Effect of Non-Uniform Delays in models of El-Niño Oscillations

Shweta Kumari

A dissertation submitted for the partial fulfilment of BS-MS dual degree in Science

Indian Institute of Science Education and Research Mohali

April 2015

Certificate of Examination

This is to certify that the dissertation titled **"Effect of Non-Uniform Delays in models of El-Niño Oscillations**

submitted by **Shweta Kumari** (Reg. No. MS10104) for the partial fulfillment of BS-MS dual degree program of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Abhishek Chaudhuri

Dr. H. K. Jassal

Dr. Sudeshna Sinha
(Supervisor)

Dated: April 23, 2015

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Dr. Sudeshna Sinha at the Indian Institute of Science Education and Research Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Shweta Kumari
(Candidate)

Dated: April 23, 2015

In my capacity as the supervisor of the candidate's project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Dr. Sudeshna Sinha
(Supervisor)

Acknowledgment

I am very thankful to Dr. Sudeshna Sinha for giving me the opportunity to learn from her. She is a wonderful guide. I am very thankful to Pranay Deep Rungta and Chandrakala Meena for their help and support throughout the project. It was a fun learning environment with these people being around. I would like to thank Pranay Deep Rungta and Imran Khan Mulani for being a moral booster. 5th year period was the most wonderful time (with the most wonderful guide and colleagues) of all the five years and I have learnt so much in it.

I am very thankful to my family for having so much faith in my capability. They have been very encouraging and supporting at every step of my life.

A big thanks to all!!

List of Figures

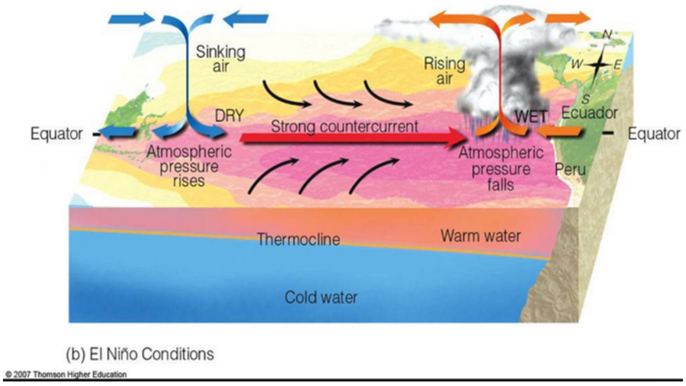
Abstract

We investigate a simple nonlinear model, modelling the El Nino/ Southern Oscillation phenomena, which arises through the strong coupling of the ocean-atmosphere system. An important feature of this class of models is the inclusion of a delayed feedback which incorporates oceanic wave transit effects, namely the effects of trapped ocean waves propagating in a basin with closed boundaries. The model allows multiple steady states. When these fixed points become unstable, one obtains self-sustained oscillations. Thus this class of models provide a simple explanation of ENSO, and provide insights on the key features that allow the emergence of oscillatory behaviour.

Contents

1. Introduction

Due to large scale interaction between ocean and atmosphere, every 3 to 7 years an event called El-Niño southern oscillator (ENSO) occurs in central and East central Pacific Ocean. Ref. [2] It brings global changes in surface temperature and rainfall. The air containing rain water disappears over north eastern Australia and the surface temperature of the sea cools down. So no further rain clouds form. The air with high temperature over the ocean is shifted to east. The winds in South America that cool the ocean get weaker, and the surface water of sea heats up. Consequently rain clouds are formed (see schematic figure). Thus the El-Niño is initiated. Ref. [5]



El-Niño refers to the name 'Christ child' as it appears around Christmas. Ref. [4] During El-Niño the southern border of Ecuador, Peru, Chile, Southern Brazil and northern Argentina faces heavy rain causing floods. It rains for more than six months in the southern border of Ecuador and average rain is over 3 m rather than its usual 20 cm. While at the same time, places near Eastern Australia face drought for 2 years or more. The temperature in these places goes up to 40 C and is a cause of forest fires. Due to this event, the macro economy of many countries are considerably affected.

2. Delay Differential Equation Model of the El-Niño Effect

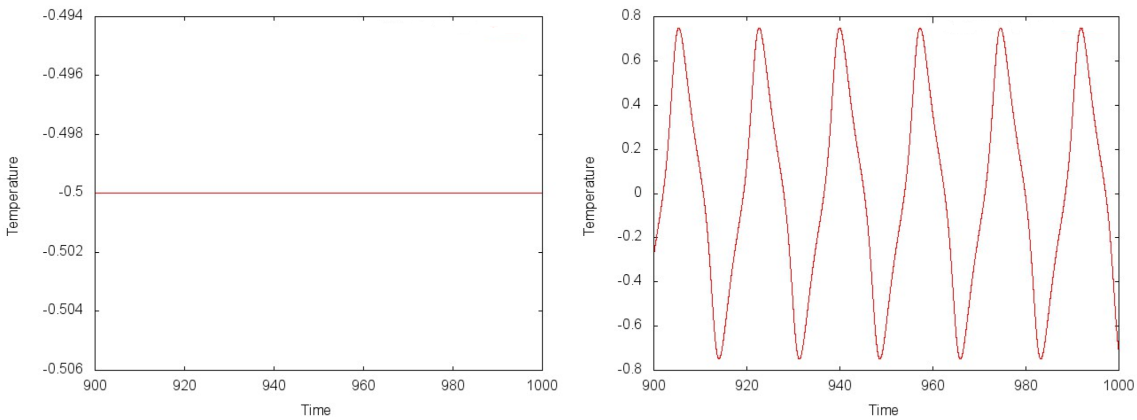
The temperature fluctuations arising in ENSO is described by the following differential equation. Ref. [6]

$$\frac{dT}{dt} = T - T^3 - \alpha T(t - \delta_s) \tag{2.1}$$

Here T is temperature anomaly in mean sea surface temperature, α is the self-coupling strength and δ_s is the (self-delay) time taken by the trapped Rossby wave to arrive at and Kelvin waves to pass the region. The system is also interesting from the general point of view of time- delayed dynamical system.

In this study we want to investigate the affect of delay on the temperature anomalies. So, we focus on the changing δ_s and keep α fixed at 0.75 in equation 2.1.

For delay $\delta_s <= 1.55$ temperature fluctuation in time always goes to a fixed point.



From the above plot it may be predicted that if the delay in equation 1 is more than 1.55 years, the temperature perturbations of the region are not stabilized and El-Niño may arrive at this equatorial Pacific region.

Python Programming language was used to study this model. The following program was written:

```
from random import *
#---------------------variables-------------------------
delta=1.56 #self delay
dt=0.01 #time step
tr=900 #transient
tf=1000 #final time
ts=100 #time step
a=0.75 #alpha

#----------initiating random temperature values----------
n=int('%0.0f'%(ts/dt))
T= [ (2*random()-1) for i in range(n)]

#------------------evolving temperature------------------

f=open('elnino␣ds=%s.txt'%delta,'w')

t=0
while t<=tr:
   i = int( '%0.0f'%(t/dt) ) %n
   j = int('%0.0f'%( ( (t-delta) /dt) -1 ) ) %n
   dT = T[i-1] - T[i-1]**3 - a*T[j]
   T[i] = T[i-1] + dT*dt
   t+=dt

t=tr
while t<=tf:
   i = int( '%0.0f'%(t/dt) ) %n
   j = int('%0.0f'%( ( (t-delta) /dt) -1 ) ) %n
   dT = T[i-1] - T[i-1]**3 - a*T[j]
   T[i] = T[i-1] + dT*dt
   f.write('%s␣\t␣%s\n'%(t,T[i]))
   t+=dt

f.close()
#-------------------------------
print 'done!'
```

3. Model of Two Coupled Regions

Here we consider two such coupled non-identical sub-systems where the temperature of the two regions are represented by following differential equations:

$$\frac{dT_1}{dt} = T_1 - T_1^3 - \alpha T_1(t - \delta_{s1}) + \gamma T_2$$
$$\frac{dT_2}{dt} = T_2 - T_2^3 - \alpha T_2(t - \delta_{s2}) + \gamma T_1$$

Here T_i $1, 2$ and δ_{si} $(i = 1, 2)$ are the temperature and self-delay of the two regions. Here γ is the coupling constant and it reflects the strength of interaction of the two regions. We are interested in finding the phenomena that arise when $\delta_{s1} \neq \delta_{s2}$, namely, the two sub-system are not identical.

Program to analyze the system is written below:

```
from random import *

def elnino(d_self1,d_self2,x):
   #-------------parameters--------------
   a=0.75
   d_coupling1=0.0
   d_coupling2=0.0
   b=0.5
   ts= 51
   tr=850
   dt=0.01
   tf=1000

   n=int('%0.0f' %(ts/dt))
   T1= [ (2*random()-1) for i in range(n)]
   T2= [ (2*random()-1) for i in range(n)]

   f = open('ds1=%s__ds2=%s__x=%s.txt' \
          %(d_self1,d_self2,x),'w')


   #--------------evolve------------
   t=0
   while t<=tr:
      c = int('%0.0f'%(t/dt)) %n
```

```
        e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
        e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
        g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
        g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

        dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
        dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]

        T1[c] = T1[c-1] + dT1*dt
        T2[c] = T2[c-1] + dT2*dt
        t+=dt

    t=tr
    while t<=tf:
        c = int('%0.0f'%(t/dt)) %n

        e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
        e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
        g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
        g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

        dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
        dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]

        T1[c] = T1[c-1] + dT1*dt
        T2[c] = T2[c-1] + dT2*dt
        f.write('%s \t %0.10f \t %0.10f\n' %(t,T1[c],T2[c]))
        t+=dt


ds1_range = [1.5+0.01*i for i in range(31)]
d_self2 = 24.0
b=0.5

for d_self1 in ds1_range:
    for x in range(1):
        elnino(d_self1,d_self2,x)

print 'done!'
```

3.1 RESULTS:

The key observations from extensive numerical simulation are as follows:

- If inter region coupling strength of the system is zero then time period of oscillation of both the oscillators are independent of each other, as expected.

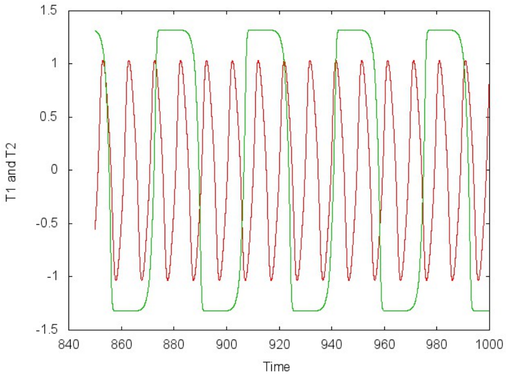


Figure 3.1: $\alpha = 0.75, \delta_{s1} = 2$, $\delta_{s2} = 15$ red color is T_1, green is T_2

- It was observed that with γ being non-zero, if temperature of one island is at fixed point, then either both system go to fixed points or go to oscillations.

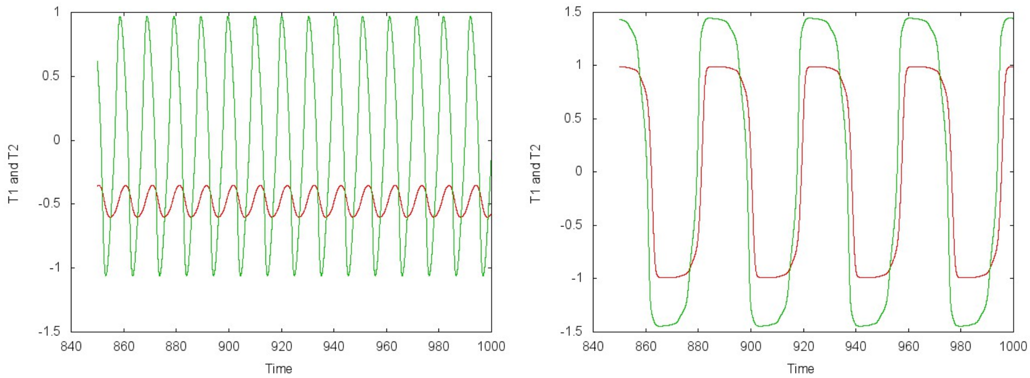


Figure 3.2: $\alpha = 0.75$: $\gamma = 0.1$ $\delta_{s1} = 0$ $\delta_{s2} = 2$(left), $\gamma = 0.2$, $\delta_{s1} = 0$ $\delta_{s2} = 15$ (right) red color is T_1, green is T_2

- For $\alpha = 0.75, \gamma = 0.2$ oscillations arose when $\delta_{s1} = \delta_{s2} = 1.9$.

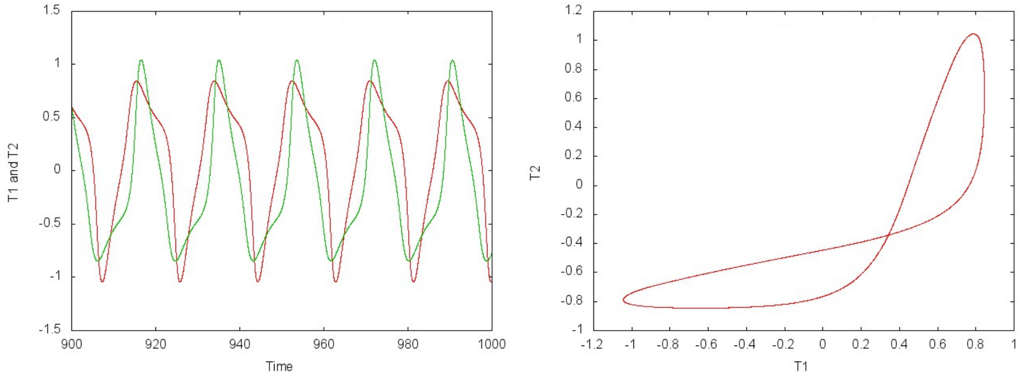


Figure 3.3: $\alpha = 0.75, \delta_{s1} = \delta_{s2} = 1.9$, red color is T_1 and green us T_2 (left), Phase portrait (right)

- The phase portrait shows that T_1 and T_2 are in phase and the oscillations are synchronized from $\alpha = 0.75$, $\gamma = 0.2$ $\delta_{s1} = \delta_{s2} = 2.0$ value.

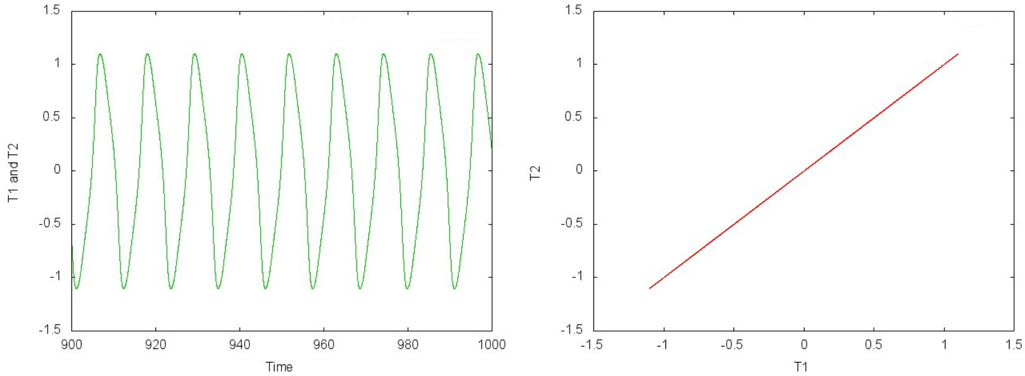


Figure 3.4: $\alpha = 0.75, \delta_{s1} = \delta_{s2} = 2.0$, red color is T_1 and green us T_2 (left), Phase portrait (right)

- If coupling γ increases, oscillations in the system arise for bigger δ_s values.
  - For $\alpha = 0.75, \delta_{s1} = 0$

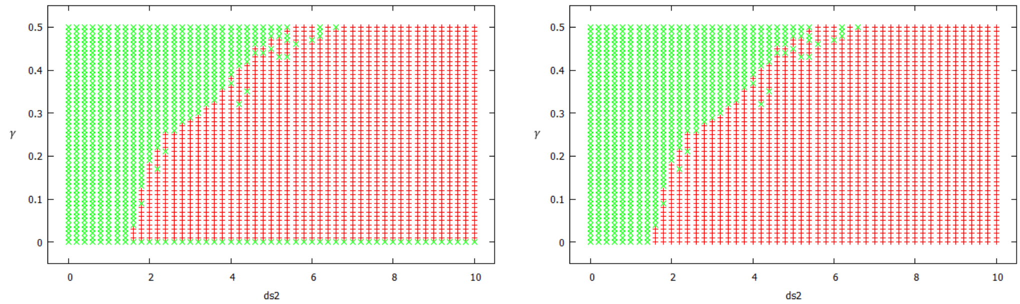


Figure 3.5: Green points represent region where one obtain fixed points and red ones are where oscillations emerge

  - For $\alpha = 0.75, \delta_{s1} = 2$

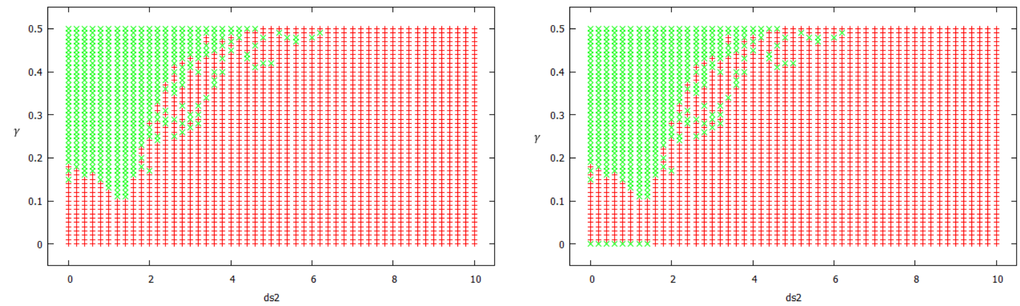


Figure 3.6: Green points represent region where one obtain fixed points and red ones are where oscillations emerge

  - For $\alpha = 0.75, \delta_{s1} = 5$

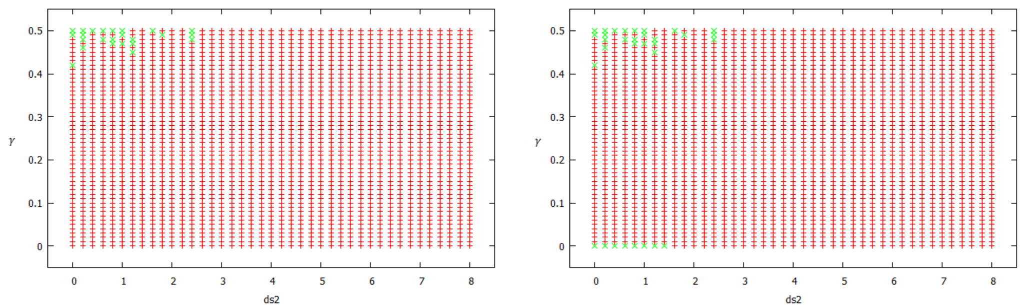


Figure 3.7: Green points represent region where one obtain fixed points and red ones are where oscillations emerge

Program to obtain the above graph is as follows:

```
from random import *

def delta_plot(d_self1,d_self2,b):
   #------------parameters--------------
   a=0.75
   ts= 20
   tr= 900
   d_coupling1 = 0.0
   d_coupling2 = 0.0
   acc1=0.0001
   acc2=0.0001
   dt=0.01
   #------------------------------------
   n=int('%0.0f' %(ts/dt))
   T1= [ (2*random()-1) for i in range(n)]
   T2= [ (2*random()-1) for i in range(n)]

   tf=1000
   m = int('%0.0f'%((tf-tr)/dt))
   dT = [0.0 for i in range(m)]
   #--------------evolve------------
   t=0
   while t<=tr:
      c = int('%0.0f'%(t/dt)) %n

      e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
      e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
      g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
      g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

      dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
      dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]

      T1[c] = T1[c-1] + dT1*dt
      T2[c] = T2[c-1] + dT2*dt
      t+=dt

   maxi1=T1[c]
   mini1=T1[c]
   maxi2=T2[c]
   mini2=T2[c]

   t=tr
   while t<=tf:
      c = int('%0.0f'%(t/dt)) %n

      e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
```

```
      e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
      g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
      g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

      dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
      dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]

      T1[c] = T1[c-1] + dT1*dt
      T2[c] = T2[c-1] + dT2*dt

      if(T1[c]>maxi1): maxi1=T1[c]
      if(T1[c]<mini1): mini1=T1[c]
      if(T2[c]>maxi2): maxi2=T2[c]
      if(T2[c]<mini2): mini2=T2[c]

      t+=dt

   if acc1>= abs(maxi1-mini1):
      c1=1
   else:
      c1=0

   if acc2>= abs(maxi2-mini2):
      c2=1
   else:
      c2=0
   return [c1,c2]
#-----------------------------------------------

#----------------main programme-----------------
d_self1= 0.0
d_range =[0.0+(0.2*i) for i in range(51)]
b_range =[0.01*i for i in range(51)]

f11=open('T1␣oscillation␣%s.txt'%d_self1,'w')
f12=open('T1␣fp␣%s.txt'%d_self1,'w')
f21=open('T2␣oscillation␣%s.txt'%d_self1,'w')
f22=open('T2␣fp␣%s.txt'%d_self1,'w')

print "total␣iterations=␣", len(d_range)*len(b_range)
count=0
for b in b_range:
   for d_self2 in d_range:
      count+=1
      print "\r␣count␣=␣%s␣\t␣b=%s␣\t␣d_self=%s␣" \
      %(count,b, d_self2) ,
      s=delta_plot(d_self1,d_self2,b)
```

```
        if s[0]==0:
            f11.write('%s␣\t␣%s␣\t␣%s␣\n' %(b,d_self2,s[0]))
        else:
            f12.write('%s␣\t␣%s␣\t␣%s␣\n' %(b,d_self2,s[0]))
        if s[1]==0:
            f21.write('%s␣\t␣%s␣\t␣%s␣\n' %(b,d_self2,s[1]))
        else:
            f22.write('%s␣\t␣%s␣\t␣%s␣\n' %(b,d_self2,s[1]))

print 'done!'

#-----------------------------------------------
```

4. Investigation of the Time Period of Oscillations

- The time period of oscillation increases with increasing δ_s value as evident from fig 3.1.

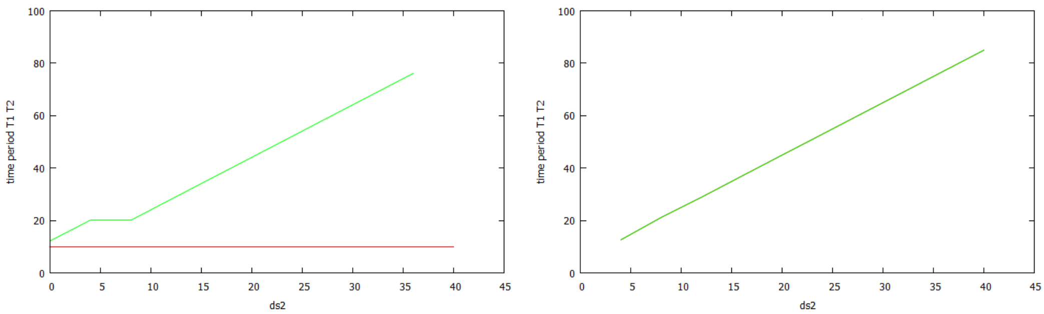


Figure 4.1: $\gamma = 0$, $\delta_{s1} = 0$ (left) and $\gamma = 0.2$, $\delta_{s1} = 0$ (right),red is the time period of T_1, green is of T_2

- If γ is nonzero, then irrespective of δ_{s1} and δ_{s2}, the period of oscillation for both islands are same. Fig 3.2 shows representative case of $\gamma = 0.2$:

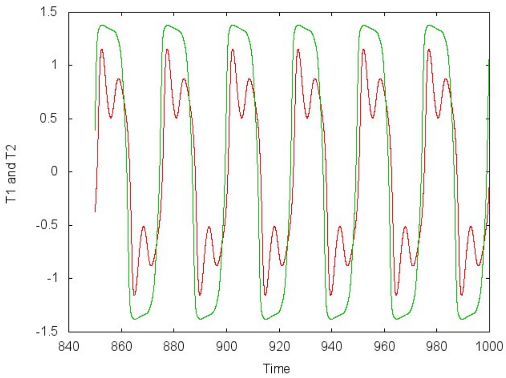


Figure 4.2: $\delta_{s1} = 2, \delta_{s2} = 10$, T_1 is red color, T_2 is green color

Programming was done in Python language. Code is given below:

```
from random import *

d_self1 = 2.0
delta = [4.0*i for i in range(11)]
f = open('T1.txt','w')
g = open('T2.txt','w')
lt1=[]
lt2=[]
j=0
while j <= len(delta)-1:
   d_self2 = delta[j]
   #-------------parameters--------------
   a=0.75
   b=0.0
   ts= 55
   dt=0.01
   d_coupling1=0.0
   d_coupling2=0.0

   n=int('%0.0f' %(ts/dt))
   T1= [ (2*random()-1) for i in range(n)]
   T2= [ (2*random()-1) for i in range(n)]

   tf=1000
   #---------------evolve------------
   t=0
   while t<=tf:
      c = int('%0.0f'%(t/dt)) %n

      e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
      e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
      g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
      g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

      dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
      dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]

      T1[c] = T1[c-1] + dT1*dt
      T2[c] = T2[c-1] + dT2*dt


      if t>=800:
         if T1[c-1]<0.0 and T1[c]>0.0:
            lt1+=[t]
            dlt1 = lt1[]-lt1
            s1+=dlt1
            c1+=1
```

```
        if T2[c-1]<0.0 and T2[c]>0.0:
            dlt2 = t-lt
            s2+=dlt2
            c2+=1
    t+=dt
f.write('\n\n')
g.write('\n\n')
j+=1
```

5. Temporal Patterns

For different values of δ_{s1}, δ_{s2} and γ, different patterns of oscillations were found. Here T_1 and T_2 have same periods of oscillations. The temperature v/s time plot with the corresponding phase portrait, are displayed below.

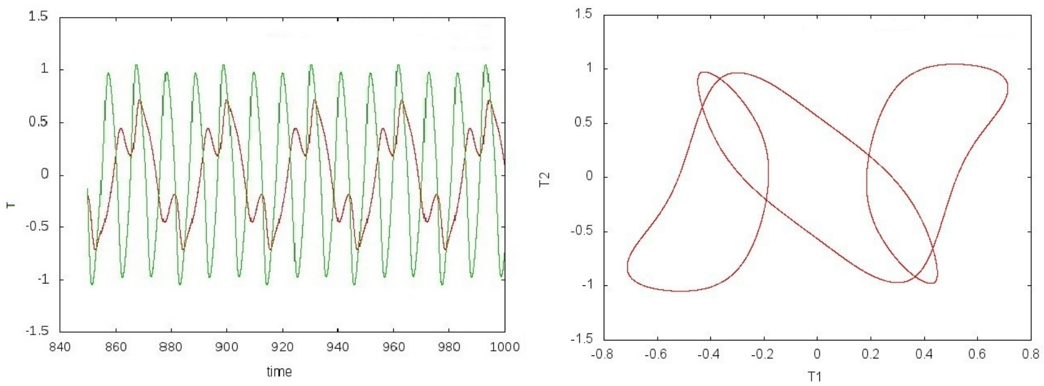


Figure 5.1: $\delta_{s1} = 1$ $\delta_{s2} = 2$ $\gamma = 0.1$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

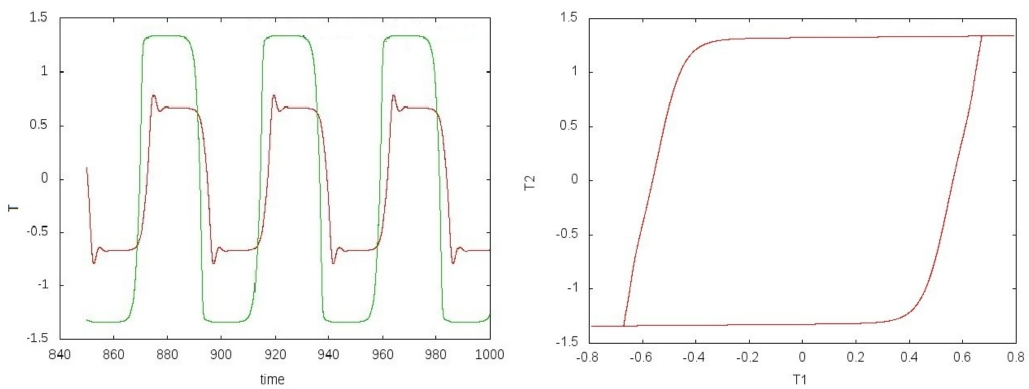


Figure 5.2: $\delta_{s1} = 1$ $\delta_{s2} = 20$ $\gamma = 0.2$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

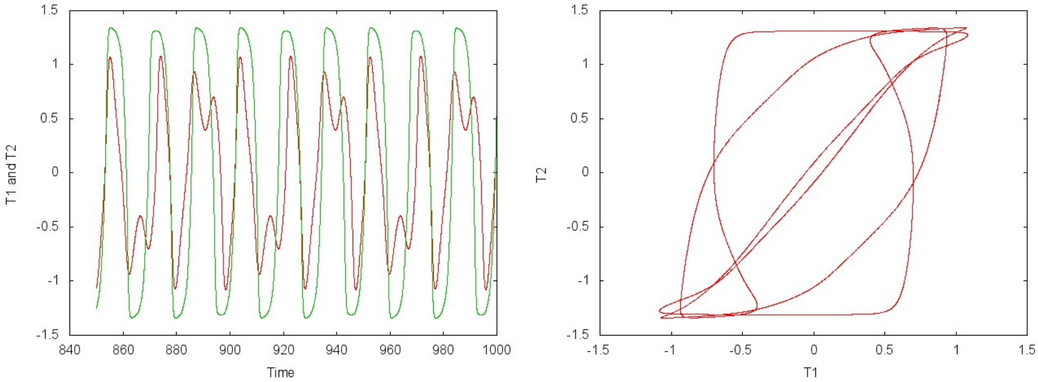


Figure 5.3: $\delta_{s1} = 2$ $\delta_{s2} = 6$ $\gamma = 0.1$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

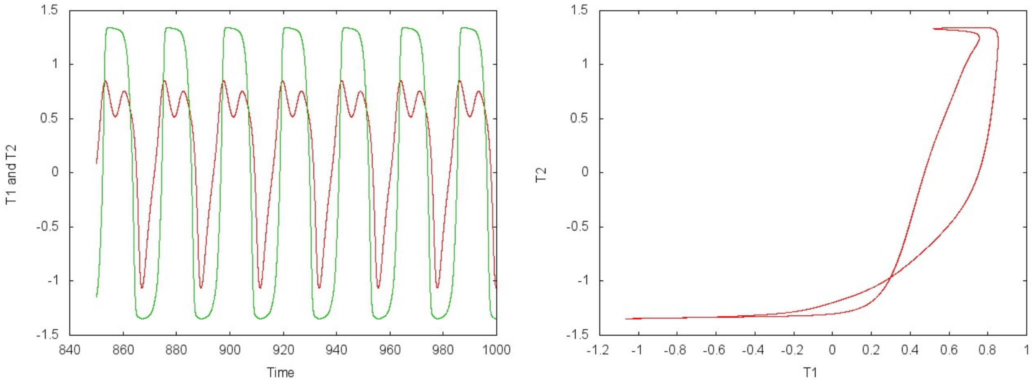


Figure 5.4: $\delta_{s1} = 2$ $\delta_{s2} = 9$ $\gamma = 0.1$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

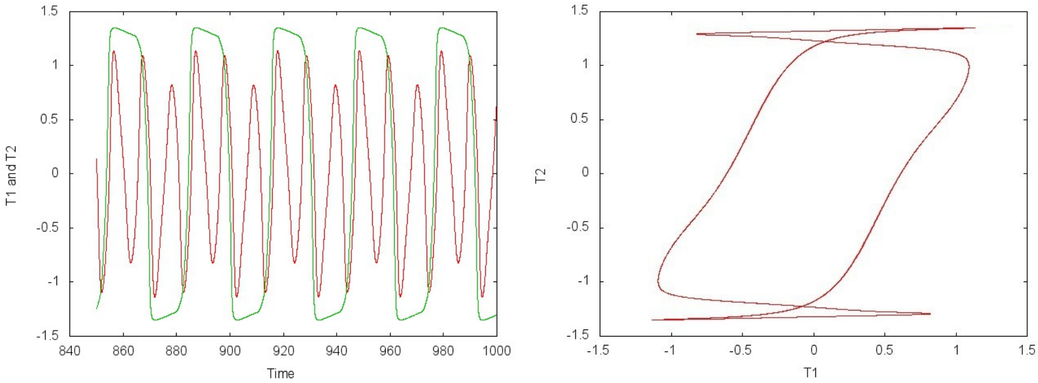


Figure 5.5: $\delta_{s1} = 2$ $\delta_{s2} = 13$ $\gamma = 0.1$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

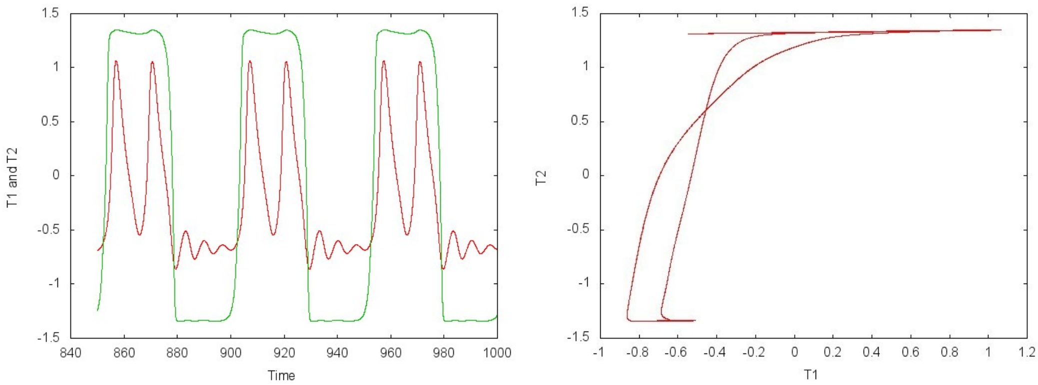


Figure 5.6: $\delta_{s1} = 2$ $\delta_{s2} = 23$ $\gamma = 0.1$: Temperature oscillation red color is T_1 and green is T_2(left), Phase portrait(right)

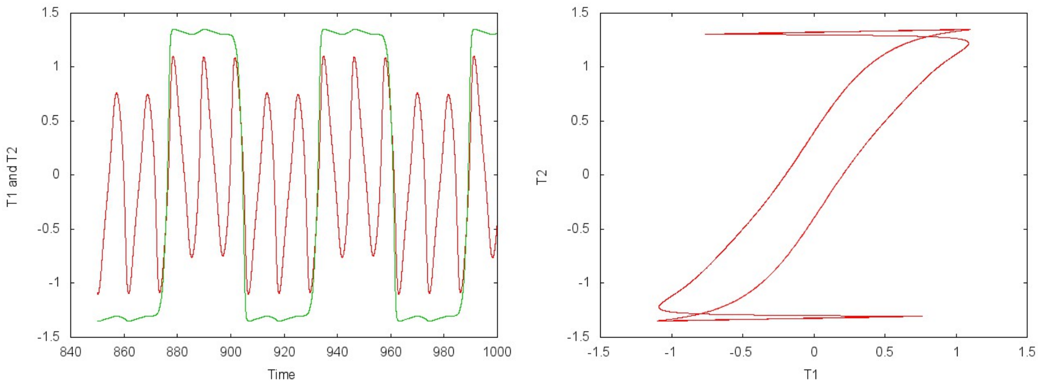


Figure 5.7: $\delta_{s1} = 2$ $\delta_{s2} = 26$ $\gamma = 0.1$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

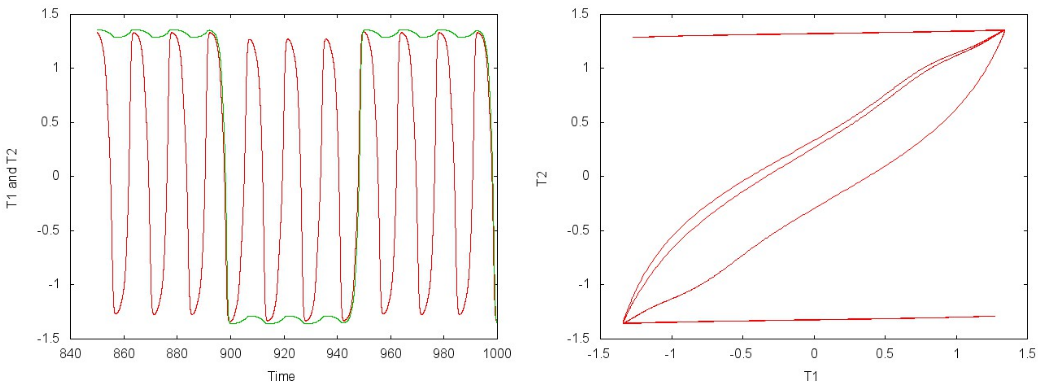


Figure 5.8: $\delta_{s1} = 5$ $\delta_{s2} = 48$ $\gamma = 0.1$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

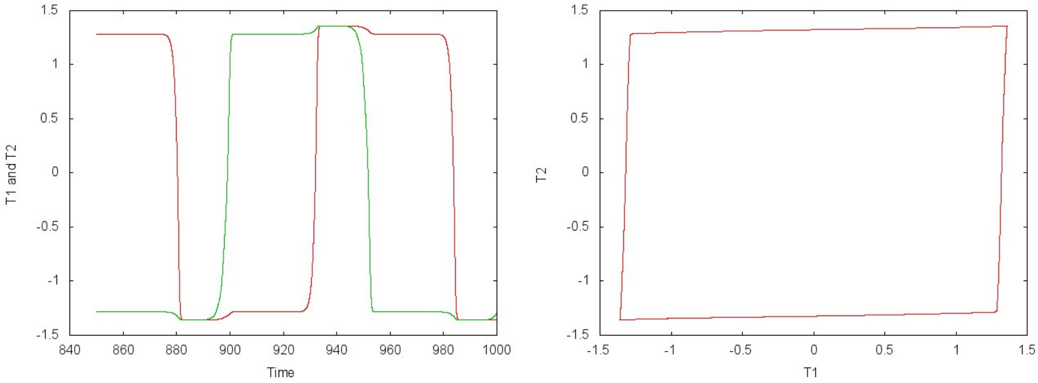


Figure 5.9: $\delta_{s1} = 50$ $\delta_{s2} = 50$ $\gamma = 0.1$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

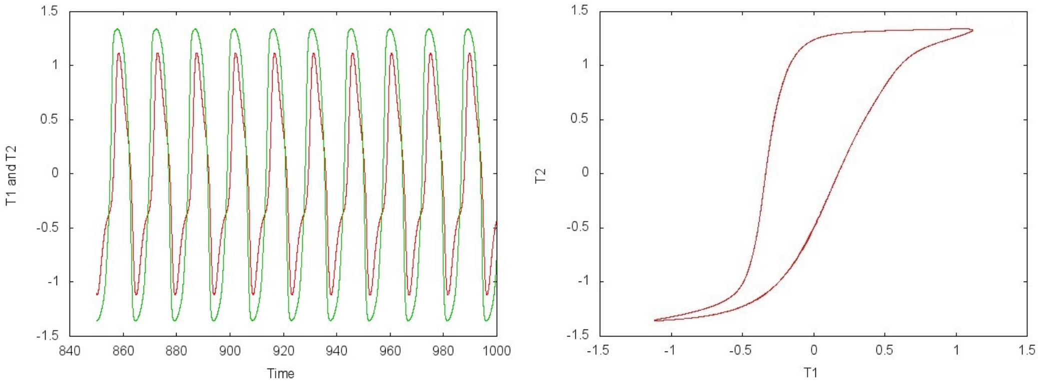


Figure 5.10: $\delta_{s1} = 2$ $\delta_{s2} = 5$ $\gamma = 0.2$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

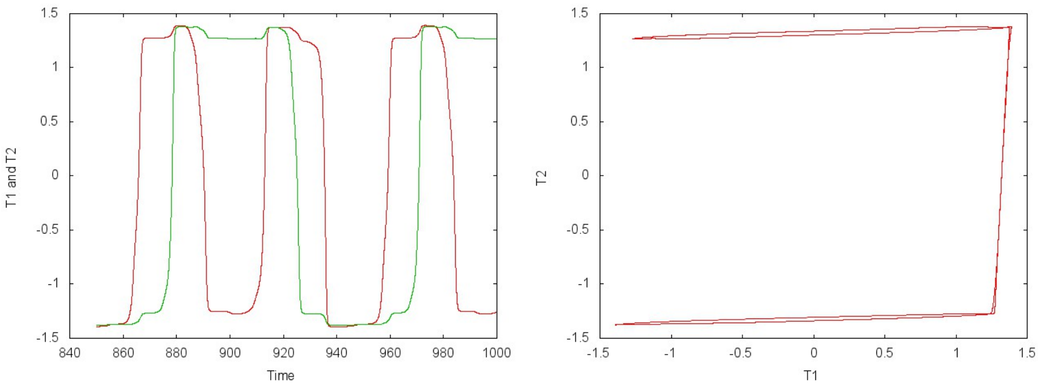


Figure 5.11: $\delta_{s1} = 21$ $\delta_{s2} = 44$ $\gamma = 0.3$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

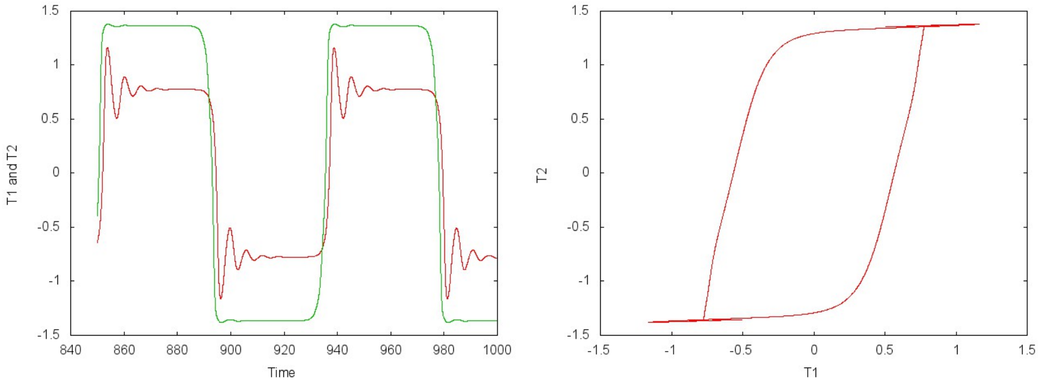


Figure 5.12: $\delta_{s1} = 2$ $\delta_{s2} = 40$ $\gamma = 0.2$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

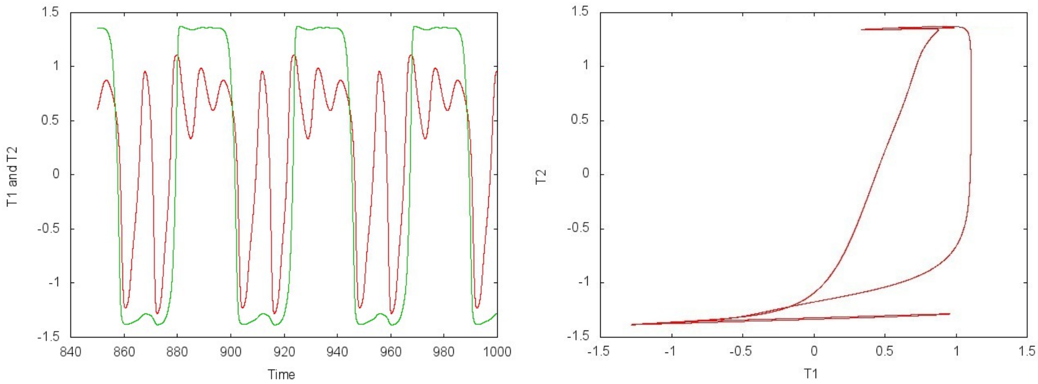


Figure 5.13: $\delta_{s1} = 3$ $\delta_{s2} = 20$ $\gamma = 0.2$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

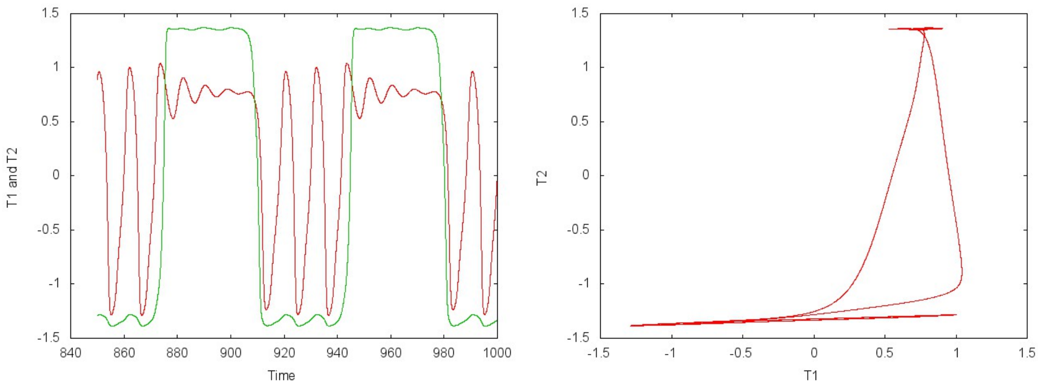


Figure 5.14: $\delta_{s1} = 3$ $\delta_{s2} = 33$ $\gamma = 0.2$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

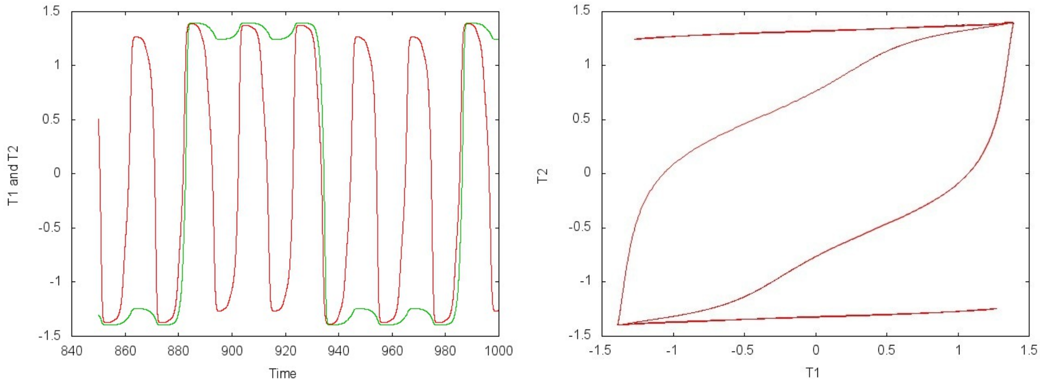


Figure 5.15: $\delta_{s1} = 8$ $\delta_{s2} = 50$ $\gamma = 0.2$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

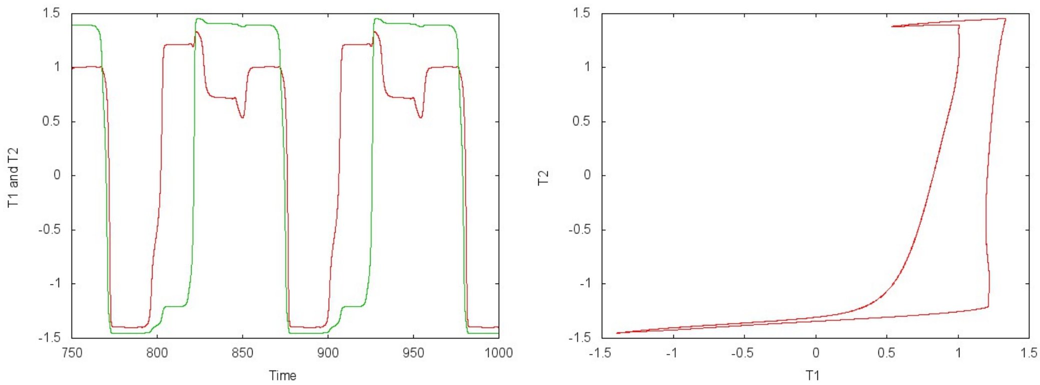


Figure 5.16: $\delta_{s1} = 21$ $\delta_{s2} = 44$ $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

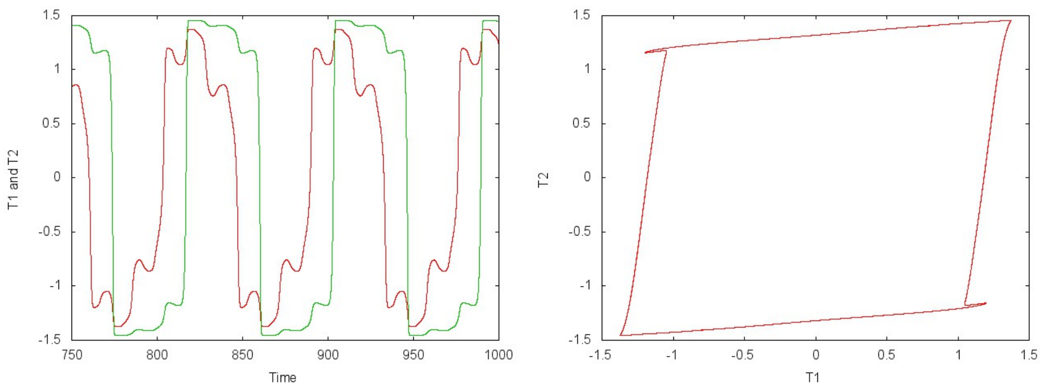


Figure 5.17: $\delta_{s1} = 24$ $\delta_{s2} = 42$ $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

If $\delta_{s1} = \delta_{s2}$, independent of the magnitude of δ_s, the temperature oscillation will have simple pattern. Both T_1 and T_2 are in synchronization. Ref. [1] (See figure 5.18).

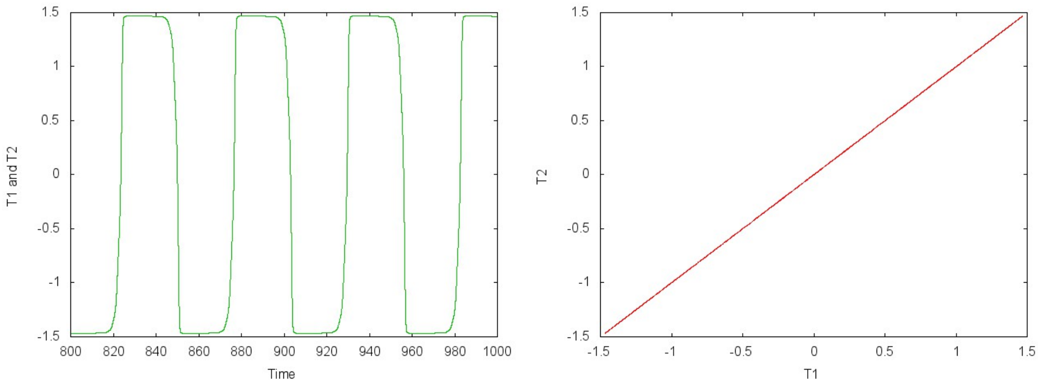


Figure 5.18: $\gamma = 0.4, \delta_{s1} = \delta_{s2} = 24$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

As an unexpected result, it was found that if $\delta_{s1} = \delta_{s2} = 50$ independent of coupling strength (γ), many of the initial conditions go to fixed points. (See figure 5.19).

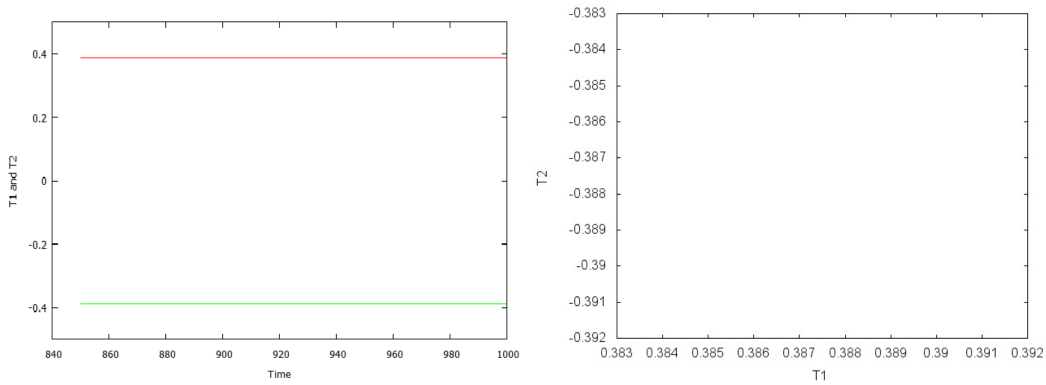


Figure 5.19: $\gamma = 0.1, \delta_{s1} = \delta_{s2} = 50$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

6. Synchronization Error

1. We observe that with increasing coupling strength, the synchronization error between T_1 and T_2 decreases. For $\delta_{s1} = 0$, as δ_{s2} increases, synchronization error between T_1 T_2 becomes zero for high values of γ.

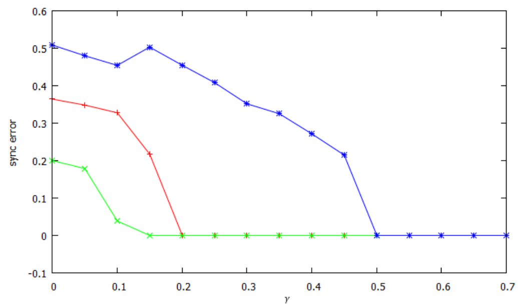


Figure 6.1: For $\delta_{s2} = 1$ (green), $\delta_{s2} = 2$ (red), $\delta_{s2} = 5$ (blue) synchronization error as a function of coupling strength γ

Program for the above plot is the following:

```
from random import *

def delta_plot(d_self1,d_self2,b):
   #-------------parameters--------------
   a=0.75
   ts= 50
   tr= 900
   d_coupling1 = 0.0
   d_coupling2 = 0.0
   dt=0.01
   #-------------------------------------
   n=int('%0.0f' %(ts/dt))
   T1= [ (2*random()-1) for i in range(n)]
   T2= [ (2*random()-1) for i in range(n)]

   tf=1000
   m = int('%0.0f'%((tf-tr)/dt))
   dT = [0.0 for i in range(m)]
   #---------------evolve------------
   t=0
   while t<=tf:
```

```
        c = int('%0.0f'%(t/dt)) %n

        e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
        e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
        g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
        g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

        dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
        dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]

        T1[c] = T1[c-1] + dT1*dt
        T2[c] = T2[c-1] + dT2*dt

        if t>=tr:
            dT[c]= abs(T1[c]-T2[c])
        t+=dt
    s=0
    j=0
    while j<=m-1:
        s+=dT[j]
        j+=1

    avg = s/m
    return avg
#---------------------------------------------

#----------------main programme---------------

d_self1= 2.0
d_self2_range= [2.0]

dT = [0.0 for i in range(10)]
b_range = [0+0.05*i for i in range(15)]

for d_self2 in d_self2_range:
    f=open('sync_b_ds1=%s␣ds2=%s.txt'%(d_self1,d_self2),'
        w')
    for b in b_range:
        i=0
        s=0
        while i<10:
            dT[i] = delta_plot(d_self1,d_self2,b)
            s+=dT[i]
            i+=1
        avg= s/10
        f.write('%s␣\t␣%s␣\n' %(b,avg))

print 'done!:D␣'
```

2. Taking average over different initial conditions we see that for different values of γ the synchronization error attains a fix valued for increasing δ_{s2}.

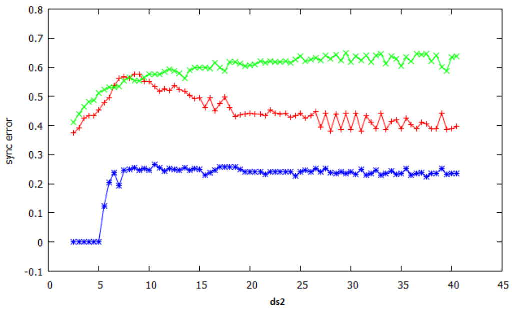


Figure 6.2: For $\delta_{s1} = 0$, synchronization error as a function of δ_{s2} : $\gamma = 0$ (green), $\gamma = 0.1$ (red), $\gamma = 0.5$ (blue)

Program for the above plot is the following:

```
from random import *

def delta_plot(d_self1,d_self2,b):
    #-------------parameters--------------
    a=0.75
    ts= 50
    tr= 900
    d_coupling1 = 0.0
    d_coupling2 = 0.0
    dt=0.01
    #-------------------------------------
    n=int('%0.0f' %(ts/dt))
    T1= [ (2*random()-1) for i in range(n)]
    T2= [ (2*random()-1) for i in range(n)]

    tf=1000
    m = int('%0.0f'%((tf-tr)/dt))
    dT = [0.0 for i in range(m)]
    #---------------evolve------------
    t=0
    while t<=tf:
        c = int('%0.0f'%(t/dt)) %n

        e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
        e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
        g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
        g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

        dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
        dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]
```

```
        T1[c] = T1[c-1] + dT1*dt
        T2[c] = T2[c-1] + dT2*dt

        if t>=tr:
           dT[c]= abs(T1[c]-T2[c])
        t+=dt
    s=0
    j=0
    while j<=m-1:
        s+=dT[j]
        j+=1

    avg = s/m
    return avg
#------------------------------------------------

#----------------main programme-----------------

d_self1= 5.0
delta = [2.0+(0.5*i) for i in range(78)]
dT = [0.0 for i in range(5)]

b_range = [0.0]

for b in b_range:
    print b,
    f=open('sync␣b=%s␣ds1=%s.txt'%(b,d_self1),'w')
    j=0
    while j<=len(delta)-2:
        d_self2 = delta[j+1]
        dd = d_self2 - d_self1

        i=0
        s=0
        while i<5:
           dT[i] = delta_plot(d_self1,d_self2,b)
           s+=dT[i]
           i+=1
        avg= s/5
        f.write('%s␣\t␣%s␣\n' %(dd,avg))
        j+=1

print 'done!'

#------------------------------------------------
```

3. **Evaluation of Basin of attraction for fixed point state. If fraction is one, fixed points are the global attractor, else if fraction is $0 < f < 1$ then we have co-existence of attractors.**

   fraction of initial conditions going to fixed points as a function of δ_{s2} is displayed in the figures below. Here we average over different initial conditions at three values of γ.

   - $\delta_{s1} = 0$

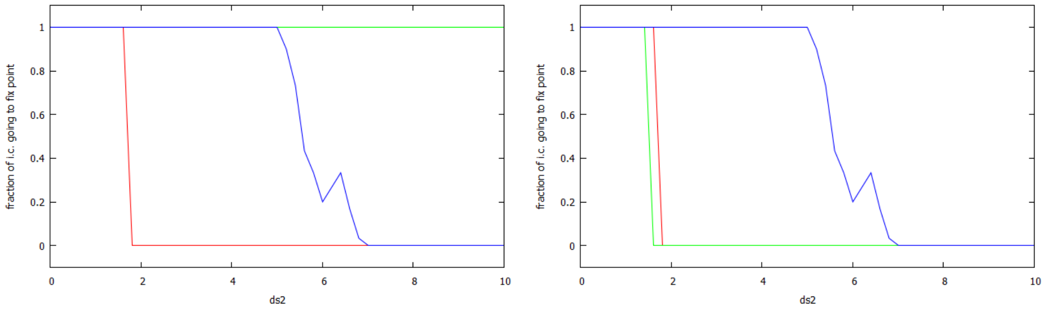


Figure 6.3: $\gamma = 0$ is green, $\gamma = 0.1$ is red, $\gamma = 0.5$ is blue, T_1 is at left, T_2 is at right

   - $\delta_{s1} = 2$

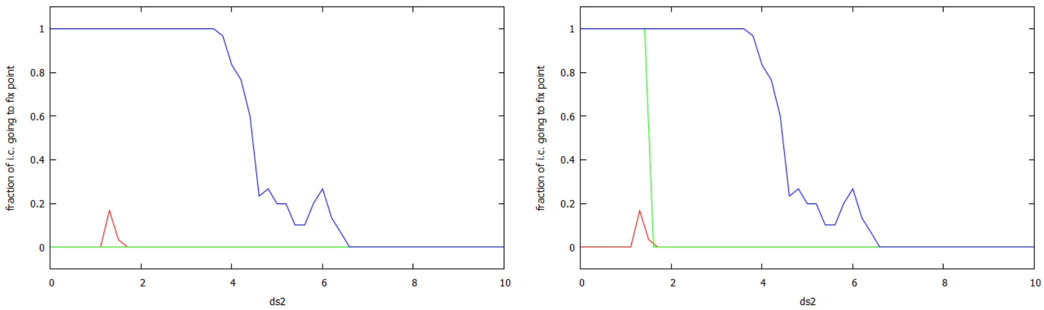


Figure 6.4: $\gamma = 0$ is green, $\gamma = 0.1$ is red, $\gamma = 0.5$ is blue, T_1 is at left, T_2 is at right

   - $\delta_{s1} = 5$

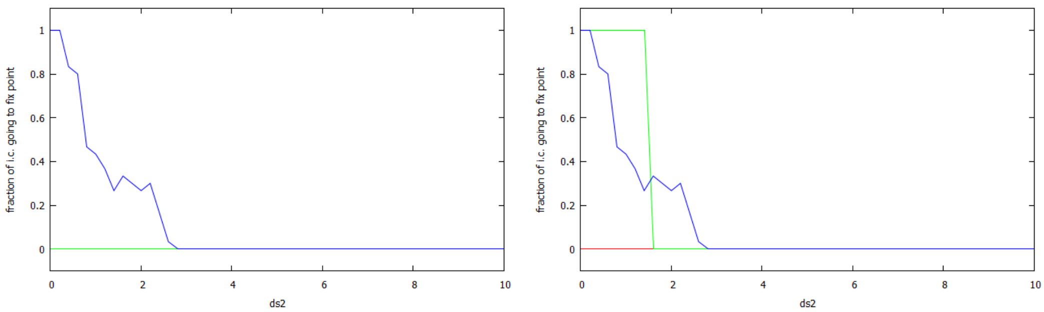


Figure 6.5: $\gamma = 0$ is green, $\gamma = 0.1$ is red, $\gamma = 0.5$ is blue, T_1 is at left, T_2 is at right

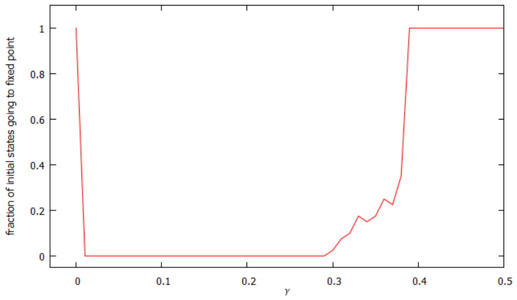
Program for the above plot is the following:

```
from random import *
import os

def delta_plot(d_self1,d_self2,b,x):
   #-------------parameters--------------
   a=0.75
   ts= 15
   tr= 900
   d_coupling1 = 0.0
   d_coupling2 = 0.0
   acc1=0.0001
   acc2=0.0001
   dt=0.01
   #-------------------------------------
   n=int('%0.0f' %(ts/dt))
   T1= [ (2*random()-1) for i in range(n)]
   T2= [ (2*random()-1) for i in range(n)]

   tf=1000
   m = int('%0.0f'%((tf-tr)/dt))
   dT = [0.0 for i in range(m)]
   #---------------evolve------------
   t=0
   while t<=tr:
      c = int('%0.0f'%(t/dt)) %n

      e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
      e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
      g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
      g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

      dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
      dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]

      T1[c] = T1[c-1] + dT1*dt
      T2[c] = T2[c-1] + dT2*dt
      t+=dt

   maxi1=T1[c]
   mini1=T1[c]
   maxi2=T2[c]
   mini2=T2[c]

   t=tr
   while t<=tf:
      c = int('%0.0f'%(t/dt)) %n
```

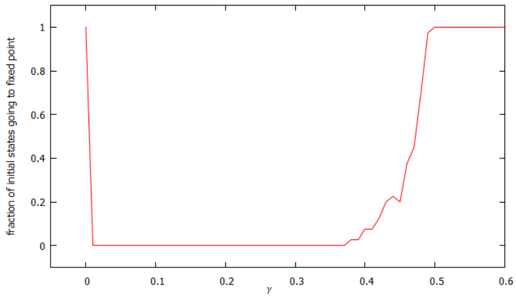
```
        e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
        e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
        g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
        g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

        dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
        dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]

        T1[c] = T1[c-1] + dT1*dt
        T2[c] = T2[c-1] + dT2*dt

        if(T1[c]>maxi1): maxi1=T1[c]
        if(T1[c]<mini1): mini1=T1[c]
        if(T2[c]>maxi2): maxi2=T2[c]
        if(T2[c]<mini2): mini2=T2[c]

        t+=dt

    if acc1>= abs(maxi1-mini1):
        c1=1
    else:
        c1=0

    if acc2>= abs(maxi2-mini2):
        c2=1
    else:
        c2=0
    return [c1,c2]
#-------------------------------------------------

#----------------main programme------------------

b_range = [0.5]
ds1_range =[5.0]
ds2_range = [0.0+(0.2*i) for i in range(16)]
for d_self1 in ds1_range:
    os.mkdir('ds1=%s'%d_self1)
    os.chdir('ds1=%s'%d_self1)
    for b in b_range:
        f=open('nfp_b=%s_ds1=%s.txt'%(b,d_self1),'w')
        for d_self2 in ds2_range:
            c=[0,0]
            i = 30
            for x in range(i):
                s = delta_plot(d_self1,d_self2,b,x)
                c[0] += s[0]
                c[1] += s[1]
```

```
            u=(1.0*c[0])/i
            v=(1.0*c[1])/i
            f.write('%s \t %s \t %s \n'%(d_self2,u,v))
        f.close()
    os.chdir('..')

#---------------------------------------
print 'done! :)'
#---------------------------------------
```

4. Basin of attraction for the fixed point: We calculate the fraction of initial condition that are attracted to fixed point as a function of coupling strength γ.



(a) T_1, $\delta_{s1} = 0$ $\delta_{s2} = 4$



(b) T_2, $\delta_{s1} = 5$ $\delta_{s2} = 2$

Program corresponding to the above graph is give below:

```
import os
from random import *

def delta_plot(d_self1,d_self2,b,x):
    #-------------parameters--------------
    a=0.75
    ts= 15
    tr= 900
    d_coupling1 = 0.0
    d_coupling2 = 0.0
    acc1=0.0001
    acc2=0.0001
    dt=0.01
    #-------------------------------------
    n=int('%0.0f' %(ts/dt))
    T1= [ (2*random()-1) for i in range(n)]
    T2= [ (2*random()-1) for i in range(n)]

    tf=1000
    m = int('%0.0f'%((tf-tr)/dt))
    dT = [0.0 for i in range(m)]
    #---------------evolve------------
    t=0
    while t<=tr:
        c = int('%0.0f'%(t/dt)) %n

        e1 = int('%0.0f'%( (t-d_self1) /dt) -1 ) ) %n
        e2 = int('%0.0f'%( (t-d_self2) /dt) -1 ) ) %n
        g1 = int('%0.0f'%( (t-d_coupling1)/dt) -1 ) ) %n
        g2 = int('%0.0f'%( (t-d_coupling2)/dt) -1 ) ) %n

        dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
        dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]
```

```
        T1[c] = T1[c-1] + dT1*dt
        T2[c] = T2[c-1] + dT2*dt
        t+=dt

    maxi1=T1[c]
    mini1=T1[c]
    maxi2=T2[c]
    mini2=T2[c]

    t=tr
    while t<=tf:
        c = int('%0.0f'%(t/dt)) %n

        e1 = int('%0.0f'%( ( (t-d_self1) /dt) -1 ) ) %n
        e2 = int('%0.0f'%( ( (t-d_self2) /dt) -1 ) ) %n
        g1 = int('%0.0f'%( ( (t-d_coupling1)/dt) -1 ) ) %n
        g2 = int('%0.0f'%( ( (t-d_coupling2)/dt) -1 ) ) %n

        dT1 = T1[c-1]-T1[c-1]**3 - a*T1[e1] + b*T2[g1]
        dT2 = T2[c-1]-T2[c-1]**3 - a*T2[e2] + b*T1[g2]

        T1[c] = T1[c-1] + dT1*dt
        T2[c] = T2[c-1] + dT2*dt

        if(T1[c]>maxi1): maxi1=T1[c]
        if(T1[c]<mini1): mini1=T1[c]
        if(T2[c]>maxi2): maxi2=T2[c]
        if(T2[c]<mini2): mini2=T2[c]

        t+=dt

    if acc1>= abs(maxi1-mini1):
        c1=1
    else:
        c1=0

    if acc2>= abs(maxi2-mini2):
        c2=1
    else:
        c2=0
    return [c1,c2]
#-------------------------------------------------

#----------------main programme------------------

b_range = [0.0+i*0.01 for i in range(51)]
ds1_range =[0.0]
```

```
ds2_range = [4.0]

for d_self1 in ds1_range:
    os.mkdir('ds1=%s'%d_self1)
    os.chdir('ds1=%s'%d_self1)
    for d_self2 in ds2_range:
        f=open('nfp__ds1=%s_ds2=%s.txt' \
             %(d_self1,d_self2),'w')
        for b in b_range:
            c=[0,0]
            i = 40
            for x in range(i):
                s = delta_plot(d_self1,d_self2,b,x)
                c[0] += s[0]
                c[1] += s[1]

            u=(1.0*c[0])/i
            v=(1.0*c[1])/i
            f.write('%s \t %s \t %s \n'%(b,u,v))
        f.close()
    os.chdir('..')

print 'done! :)'
#------------------------------------------
```

7. Alternate Model For El-Niño Effect

Here we consider two coupled non-identical sub-systems where the temperature of the two regions are represented by following differential equations:

$$\frac{dT_1}{dt} = T_1 - T_1^3 - \alpha T_1(t - \delta_{s1}) + \gamma T_2(t - \delta_{c1})$$
$$\frac{dT_2}{dt} = T_2 - T_2^3 - \alpha T_2(t - \delta_{s2}) + \gamma T_1(t - \delta_{c2})$$

Here T_i, δ_{si} and δ_{ci}, $(i = 1, 2)$ are the temperature self-delay and coupling delay of the two regions. The δ_{ci}, $(i = 1, 2)$ term represents the inter-feedback from the temperature history of the other region.

Program to analyze the system is written below:

```
from random import *

delta1=1.6 #self delay of first island
delta2=1.6 #self delay of second island
dt=0.01 #time step
tf=80 #final time
a=0.75 #alpha
b=0.5 #gamma

c1=int('%0.0f'%(delta1/dt)) - 1
c2=int('%0.0f'%(delta2/dt)) - 1
n1=c1+1
n2=c2+1

T1=[0 for i in range(n1)]
T2=[0 for i in range(n2)]
L1=open('elnino␣T1%s.txt'%delta1,'w')
L2=open('elnino␣T2%s.txt'%delta2,'w')

t=-delta2
while t<=0:
    i1=int('%0.0f'%(t/dt)) %n1
    i2=int('%0.0f'%(t/dt)) %n2

    T1[i1] = (2*random()-1)
    T2[i2] = (2*random()-1)
    L1.write('%s\t%s\n'%(t,T1[i1]))
```

```
    L2.write('%s\t%s\n'%(t,T2[i2]))
    t+=dt

t=0
while t<=tf:
    i1 = int( '%0.0f'%(t/dt) ) %n1
    i2 = int( '%0.0f'%(t/dt) ) %n2
    dT1 = T1[i1-1] - T1[i1-1]**3 - a*T1[i1] + b*T2[i2]
    dT2 = T2[i2-1] - T2[i2-1]**3 - a*T2[i2] + b*T1[i1]

    T1[i1] = T1[i1-1] + dT1*dt
    T2[i2] = T2[i2-1] + dT2*dt

    L1.write('%s \t %s \n'%(t,T1[i1]))
    L2.write('%s \t %s \n'%(t,T2[i2]))

    t+=dt

L1.close()
L2.close()
#--------------------------------
print 'done!'
```

We observed that for the uncoupled system ($\gamma = 0$) the two subsystems go to fixed point if the value of $\delta_{si} < 1.6$. Oscillation in the temperature appears for $\delta_{si} > 1.6$ values. When $\delta_{c1} = \delta_{s2}$ and $\delta_{c2} = \delta_{s1}$, coupled systems show the behavior displayed below:

- System is always in anti-phase synchronization. Ref. [3, 7]

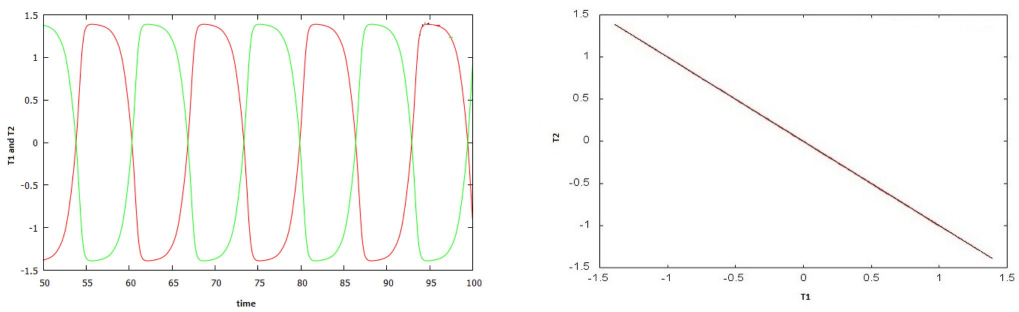
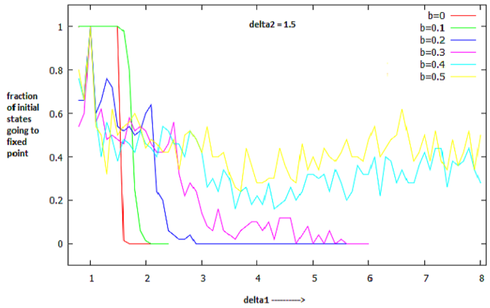


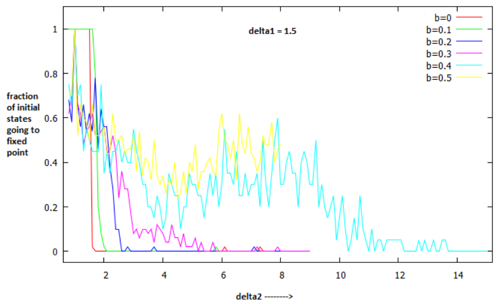
Figure 7.1: $\delta_{si} = \delta_{ci} = 2.0, (i = 1, 2)$ Red color represents T_1 and Green color represents T_2

- A system with low coupling strength and $\delta_{s1} = 1.6$, $\delta_{s2} = 0.8$ primarily attracts to a fixed point. With increasing coupling strength, oscillation arise in the subsystems for some initial conditions.

- For $\delta_{s1} = 1.6$, $\delta_{s2} = 0.8$, $\gamma = 0.1$ all initial conditions go to fixed points. For $\gamma = 0.5$, for both T_1 and T_2 we find that 58 to 66% of the initial conditions go to fixed points while the rest are attracted to cycles.
- For $\delta_{s1} = 1.6$, $\delta_{s2} = 1.6$, and $\gamma = 0.1$ both T_1 and T_2 shift towards oscillations with the increasing value of time delay.

Evaluation of Basin of attraction for fixed point state. If fraction is one, fixed points are the global attractor, else if fraction is $0 < f < 1$ then we have co-existence of attractors.





Code for the above plots is as follows:

```
from random import random
import os
os.mkdir('observations')
os.chdir('observations')

#-------------function--------------

def timeEvolution(x,delta2):
   dt=0.01
   a=0.75
   b=0.0
   acc1=0.0001
   acc2=0.0001
   tf=200.0

   n1= int('%0.0f'%(delta1/dt) )
   n2= int('%0.0f'%(delta2/dt) )

   T1=[0 for i in range(n1)]
   T2=[0 for i in range(n2)]

   f1=open('data1_%s_%s.txt'%(x,delta1),'w')
   f2=open('data2_%s_%s.txt'%(x,delta2),'w')

   t=-delta2
   while t<=0.0:
      i1= int( '%0.0f'%(t/dt) ) %n1
```

```
      i2= int( '%0.0f'%(t/dt) ) %n2
      T1[i1]= (2*random()-1)
      T2[i2]= (2*random()-1)
      f1.write('%0.2f␣\t␣%s\n'%(t,T1[i1] ))
      f2.write('%0.2f␣\t␣%s\n'%(t,T2[i2] ))
      t+= dt

   t=0.0
   while t<=tf:
      i1= int( '%0.0f'%(t/dt) ) %n1
      i2= int( '%0.0f'%(t/dt) ) %n2
      DT1 = T1[i1-1] - T1[i1-1]**3 - a*T1[i1] + b*T2[i2]
      DT2 = T2[i2-1] - T2[i2-1]**3 - a*T2[i2] + b*T1[i1]
      T1[i1] = T1[i1-1] + DT1*dt
      T2[i2] = T2[i2-1] + DT2*dt
      f1.write('%0.2f␣\t␣%0.10f␣\t%s\n'%(t,T1[i1],i1 ))
      f2.write('%0.2f␣\t␣%0.10f␣\t%s\n'%(t,T2[i2],i2 ))
      t+=dt

   p1=int( '%0.0f'%(tf/dt) )%n1
   q1=int( '%0.0f'%( (tf-100*dt)/dt ) )%n1

   p2=int( '%0.0f'%(tf/dt) )%n2
   q2=int( '%0.0f'%( (tf-100*dt)/dt ) )%n2

   f1.close()
   f2.close()

   if acc1>= abs(T1[p1]-T1[q1]):
      c1=1
   else:
      c1=0

   if acc2>= abs(T2[p2]-T2[q2]):
      c2=1
   else:
      c2=0
   return [c1,c2]

#-------parameters of main program--------
d2i=0.8
d2f=1.8
dd2=0.1
iterate=5
#------------------------
count=int((d2f-d2i)/dd2)*iterate
print "total␣iterations=%s␣\n"%count
count=0
```

```
f=open('nfp.txt','w')
for delta1 in [0.8]:
    delta2=d2i
    while delta2 <= d2f:
        os.mkdir('delta1=%s␣delta2=%s'%(delta1,delta2))
        os.chdir('delta1=%s␣delta2=%s'%(delta1,delta2))

        c=[0,0]
        for x in range(iterate):
            s=timeEvolution(x,delta2)
            c[0] += s[0]
            c[1] += s[1]

            count+=1
            print '\r␣count=%s␣' %count ,

        f.write('%s␣\t␣%s␣\t␣%s␣\t␣%s␣\n' \
             %(delta1,delta2,c[0],c[1]))
        os.chdir('..')
        delta2+=dd2

raw_input()
```

For different values of δ_{s1}, δ_{s2}, δ_{c1}, δ_{c2} and γ, different patterns of oscillations were found. Here T_1 and T_2 have same periods of oscillations. The temperature v/s time plot with the corresponding phase portrait, are displayed below.

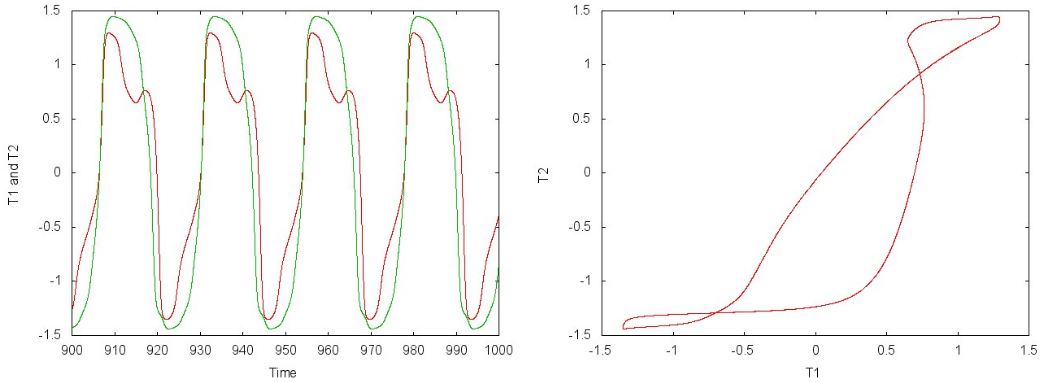


Figure 7.2: $\delta_{s1} = 4$ $\delta_{s2} = 9$ $\delta_{c1} = 0$ $\delta_{c2} = 1$ and $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

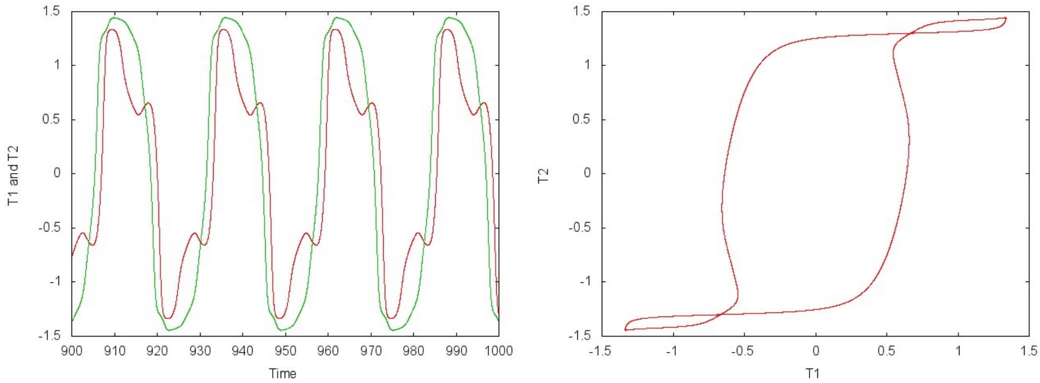


Figure 7.3: $\delta_{s1} = 4$ $\delta_{s2} = 10$ $\delta_{c1} = 0$ $\delta_{c2} = 1$ and $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

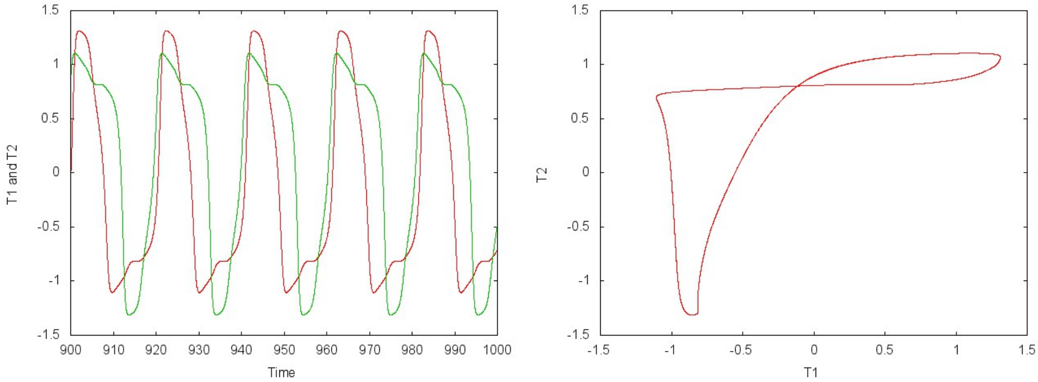


Figure 7.4: $\delta_{s1} = 4$ $\delta_{s2} = 4$ $\delta_{c1} = 0$ $\delta_{c2} = 3$ and $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

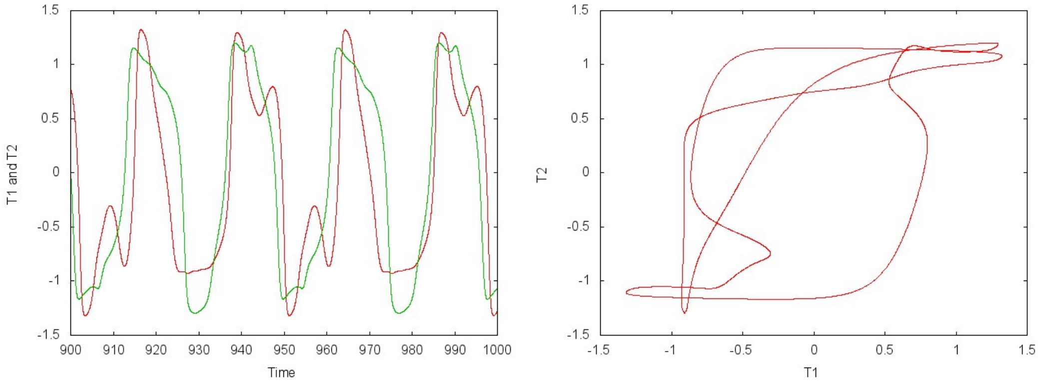


Figure 7.5: $\delta_{s1} = 3$ $\delta_{s2} = 6$ $\delta_{c1} = 0$ $\delta_{c2} = 4$ and $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

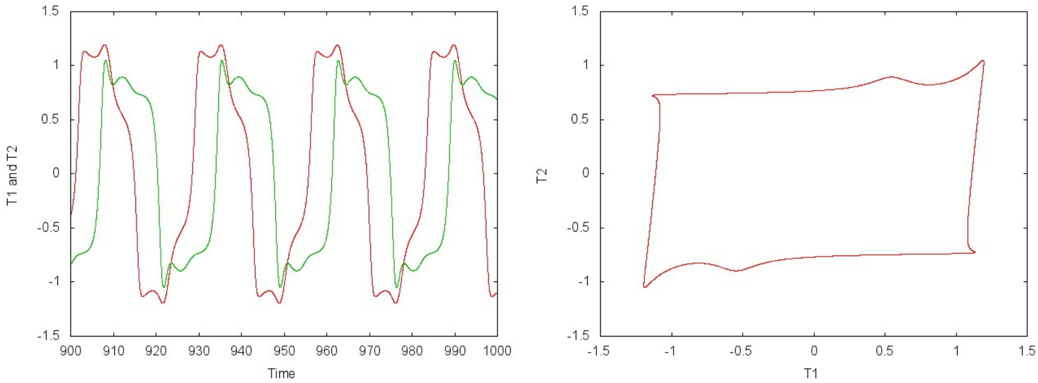


Figure 7.6: $\delta_{s1} = 7$ $\delta_{s2} = 1$ $\delta_{c1} = 0$ $\delta_{c2} = 4$ and $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

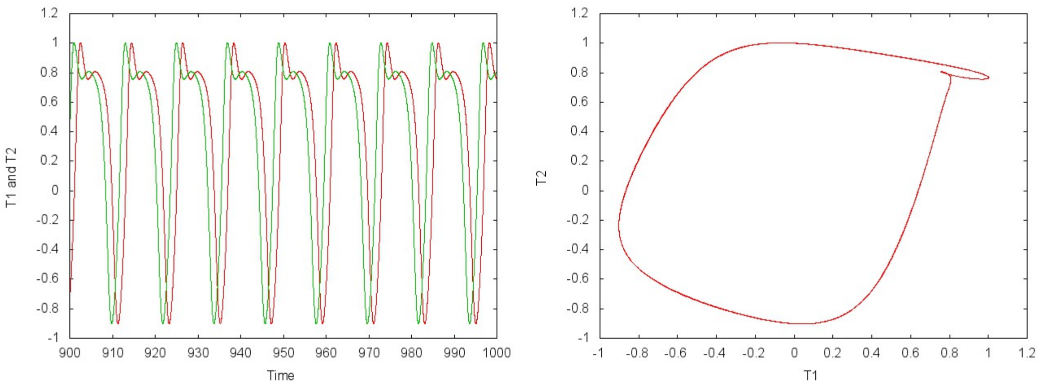


Figure 7.7: $\delta_{s1} = 1$ $\delta_{s2} = 1$ $\delta_{c1} = 0$ $\delta_{c2} = 9$ and $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

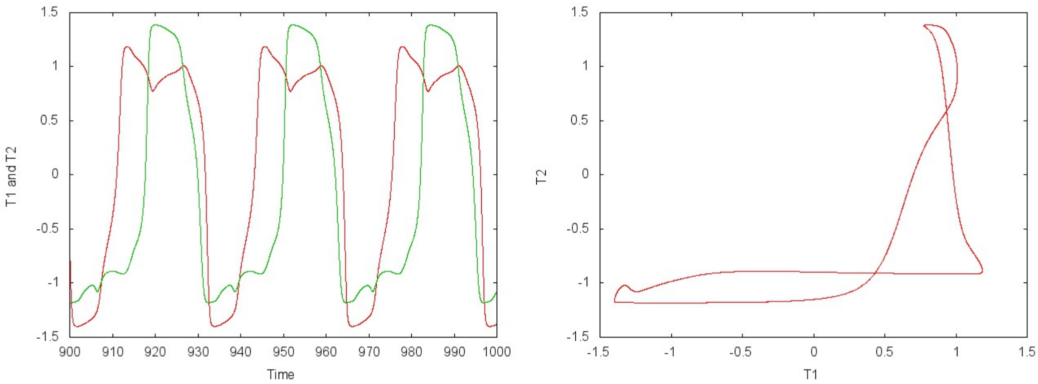


Figure 7.8: $\delta_{s1} = 7$ $\delta_{s2} = 8$ $\delta_{c1} = 1$ $\delta_{c2} = 6$ and $\gamma = 0.5$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

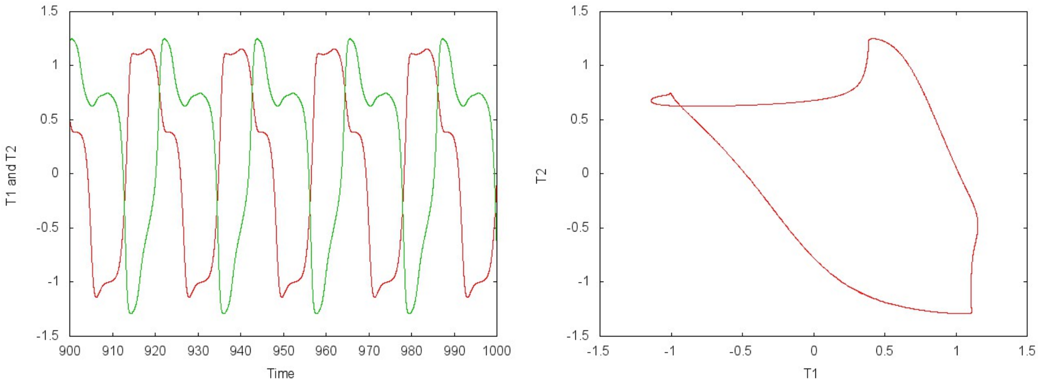


Figure 7.9: $\delta_{s1} = 7$ $\delta_{s2} = 3$ $\delta_{c1} = 1$ $\delta_{c2} = 7$ and $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

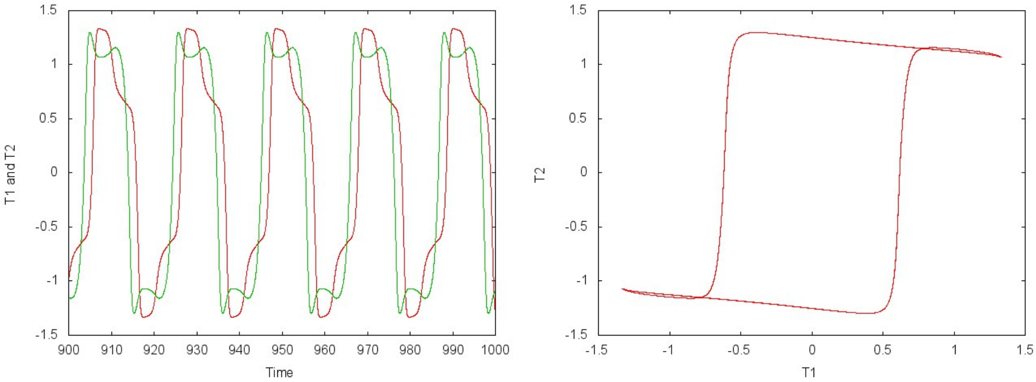


Figure 7.10: $\delta_{s1} = 4$ $\delta_{s2} = 9$ $\delta_{c1} = 1$ $\delta_{c2} = 10$ and $\gamma = 0.5$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

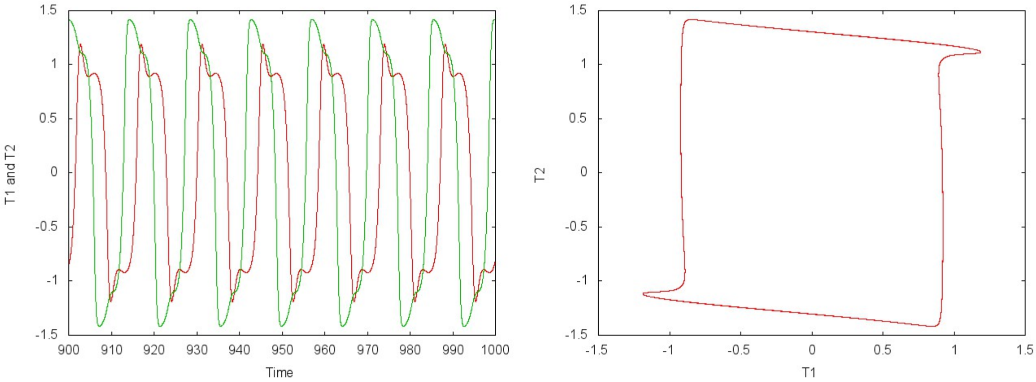


Figure 7.11: $\delta_{s1} = 1$ $\delta_{s2} = 6$ $\delta_{c1} = 2$ $\delta_{c2} = 7$ and $\gamma = 0.5$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

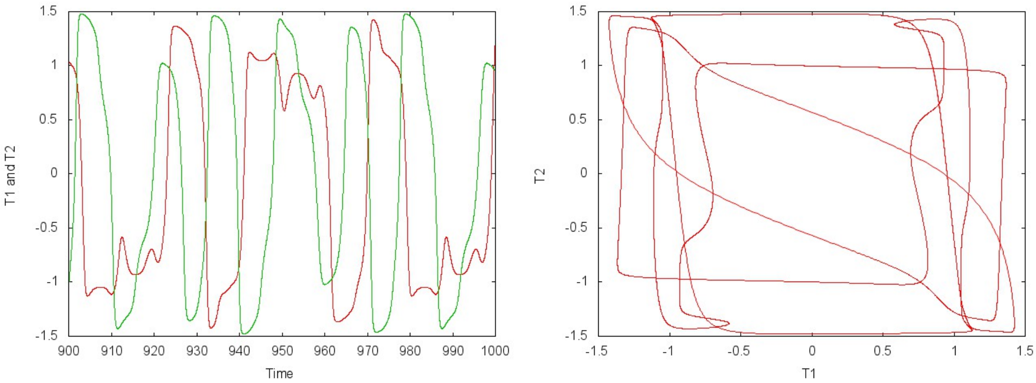


Figure 7.12: $\delta_{s1} = 8$ $\delta_{s2} = 5$ $\delta_{c1} = 2$ $\delta_{c2} = 7$ and $\gamma = 0.5$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

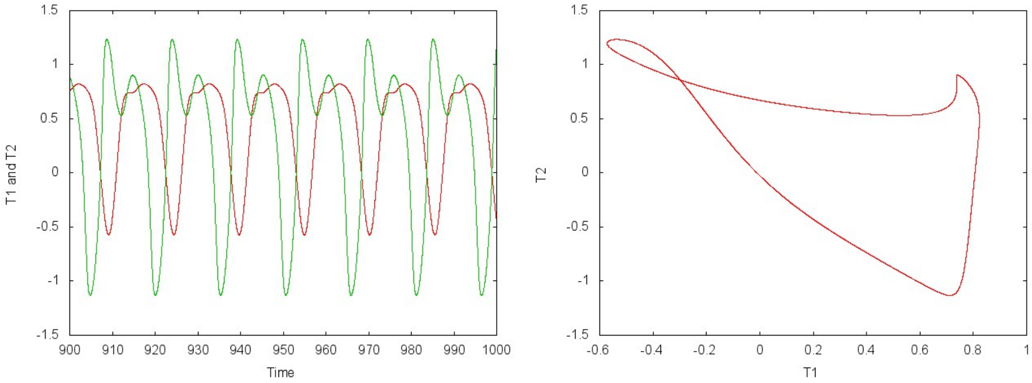


Figure 7.13: $\delta_{s1} = 0$ $\delta_{s2} = 2$ $\delta_{c1} = 2$ $\delta_{c2} = 10$ and $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

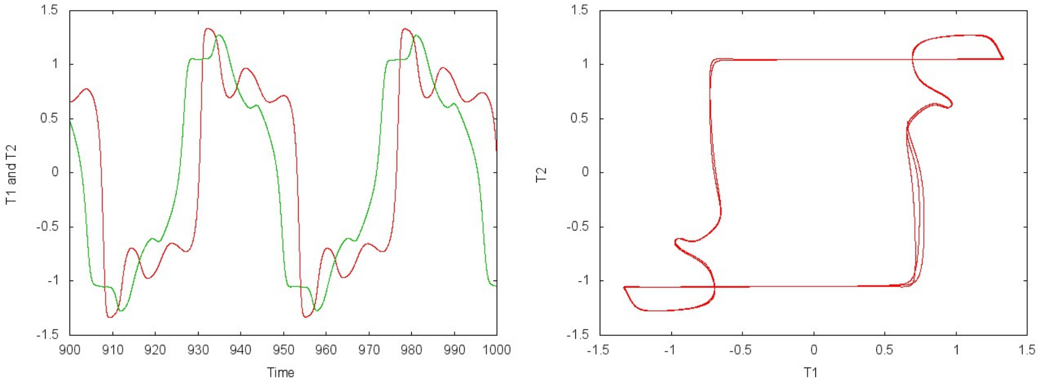


Figure 7.14: $\delta_{s1} = 4$ $\delta_{s2} = 10$ $\delta_{c1} = 3$ $\delta_{c2} = 3$ and $\gamma = 0.5$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

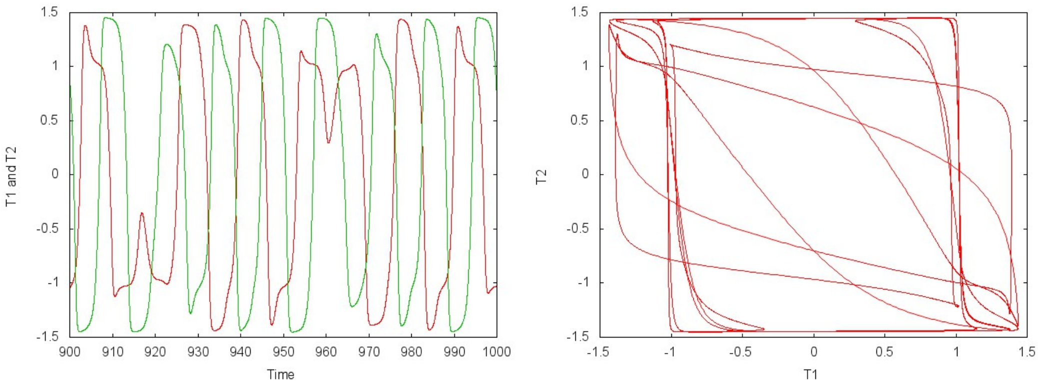


Figure 7.15: $\delta_{s1} = 6$ $\delta_{s2} = 5$ $\delta_{c1} = 3$ $\delta_{c2} = 3$ and $\gamma = 0.5$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

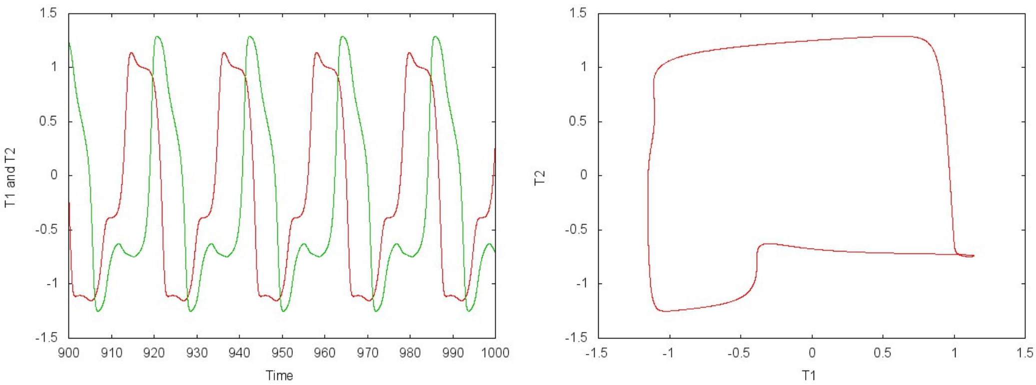


Figure 7.16: $\delta_{s1} = 7$ $\delta_{s2} = 3$ $\delta_{c1} = 3$ $\delta_{c2} = 5$ and $\gamma = 0.4$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

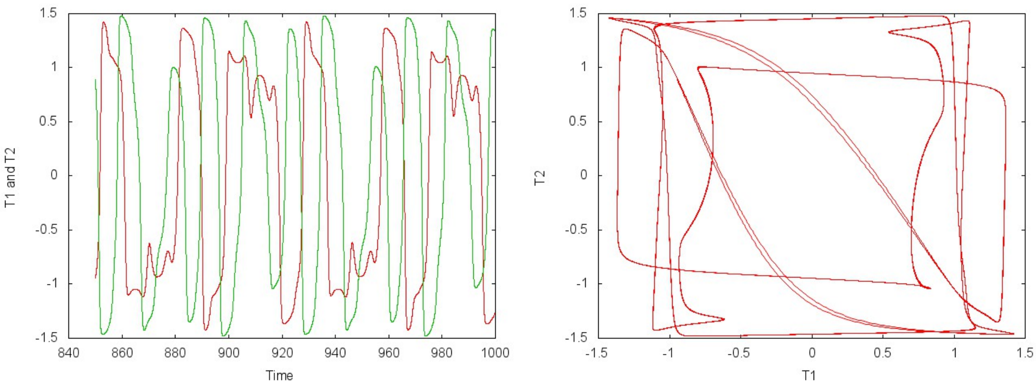


Figure 7.17: $\delta_{s1} = 8$ $\delta_{s2} = 5$ $\delta_{c1} = 3$ $\delta_{c2} = 6$ and $\gamma = 0.5$: Evolution of Temperature fluctuations: Red color representing T_1 and green T_2 in left, Phase portrait in T_1-T_2 is displayed on the right

8. Conclusion

The emergence of oscillations in models of the El-Niño effect is of utmost relevance. Here we study the prevalence of oscillations in a system of coupled non-identical delayed action oscillators modelling ENSO. We show how the non-uniformity in the delays in the sub-systems affect the rise of oscillations. We find the basin of attraction for the steady state vis-a-vis the oscillatory state of the two sub-systems. It is evident that there are regimes where the oscillatory sub-system induces oscillations in sub-system that would have gone to a steady state if uncoupled. We also find that there exists a window of intermediate coupling strengths, in certain parameter regimes, that gives oscillations. Namely, when the non-identical sub-systems are too strongly or too weakly coupled one obtains steady states, while moderate coupling allows oscillations to emerge. These results are of interest, as geographical sub-systems are most likely to be non-identical and so it is important to understand the effect of non-homogeneity in the emergence of oscillations.

Bibliography

[1] A Arenas, A Diaz-Guilera, J Kurths, Y Moreno, C Zhou. “Synchronization in complex networks” *Physics Reports*, **469, 3,(2008)**

[2] An introduction to atmospheric physics, Andrews, David G, *Cambridge University Press, 2nd edition.* **2010**

[3] A. Pikovsky, M. Rosenblum, & J Kurths “Synchronization” *Cambridge University Press, Cambridge, UK* , **2001**

[4] Encyclopedia of atmospheric sciences, North,Gerald R. (Ed.), Pyle, John (Ed.), Zhang, Fuqing (Ed.), *Amsterdam Academic Press, 2nd edition.* **2003**

[5] James R.Holton, Gregory J. Hakim. An introduction to Dynamic Meteorology. *Academic Press, 5th edition.* **2013**

[6] Ian Boutle, Richard H.S. Taylor, and Rudolf A. Rmer, *American Journal of Physics*, **January 2007 -75, Issue 1, pp. 15.**

[7] Steven Strogatz. “Nonlinear dynamics and Chaos” *ISBN*, **978-0-201-54344-5 (1994)**