El Niño Southern Oscillations (ENSO)

UBC Math 215/255, 2024WT1. S. Bachmann & K. Dao Duc
P. Harrington and N. Bailey

1 A dynamical system for El Niño

El Niño is a quasi-periodic phenomena in which the sea surface temperature (SST) becomes much colder than usual in the tropical Eastern Pacific. In fact, there are oscillations between El Niño and La Niña, the latter corresponding to a warmer than average SST in the tropical Eastern Pacific. One of the first models to describe the El Niño system considers the effect of trade winds on the interaction between the SST and the depth of the transition layer (called the thermocline) between the warm surface and the cold ocean floor. The model is summarized in the following system of ODEs:

$$\begin{cases} x' &= -x + \gamma(bx + y) - \varepsilon(bx + y)^3 \\ y' &= -ry - \alpha bx \end{cases} \tag{1}$$

where x is proportional to the *SST anomaly* in the Eastern Pacific (namely the difference between the warm SST during El Niño and standard SST), and y is proportional to the *thermocline depth anomaly* in the Western Pacific. All geophysical parameters $\gamma, b, \epsilon, r, \alpha$ are strictly positive.

The purpose of this module is to study the oscillatory solutions of the system (1), both in the linear and non-linear regimes.

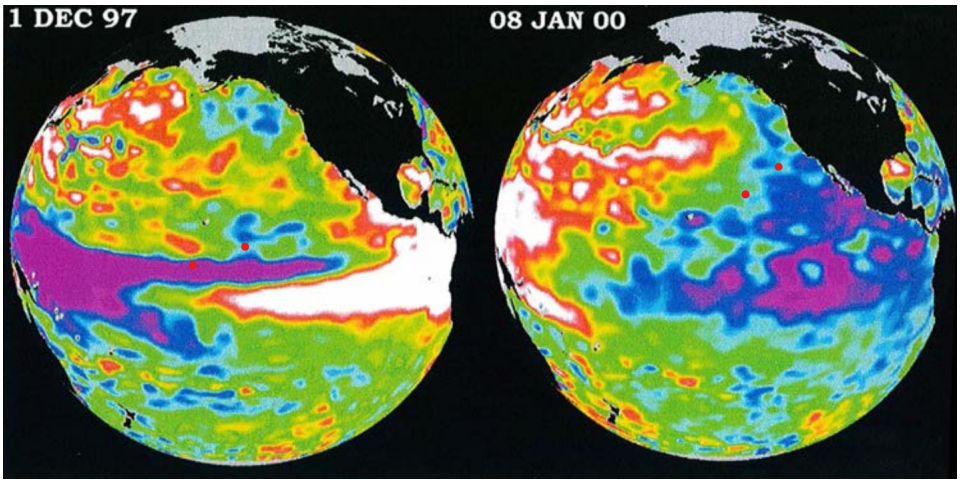


Figure 1: Temperature measurements illustrating the oscillation between El Niño (left) and La Niña (right) (Kapler and Engler, reprinted from NASA/JPL-Caltech)

2 The linearized system and its oscillatory solutions

1. Assume that $\alpha < r$ and that $\gamma b(1 - \frac{\alpha}{r}) < 1$. Find the unique steady state of (1). What geophysical situation does it describe?
2. Linearize the system about the steady state.
3. Show that the solution of the linear system is purely oscillatory if the parameters of the model are such that

$$\gamma b = r + 1 \quad \text{and} \quad -r^2 + \alpha r + \alpha > 0. \tag{2}$$

4. Let us denote $\lambda_{\pm} = \pm i\omega$ with $\omega = \sqrt{\text{Det}(A)}$. Find the general solution of (1) in the purely oscillatory case found above.

5. In physical units where time is measured in units of two months, we have that $r = \frac{1}{4}, \alpha = \frac{1}{8}$. According to the model above, what is the time between two El Niño events?

6. The constant γ is related to upwelling: Winds blowing across the ocean surface push water away, resulting in colder water rising up from beneath the surface. Assume that γ increases slightly from the value above, all other parameters staying constant. Describe qualitatively the behaviour of the solution of the ENSO system.

3 Numerical exploration (Python)

We first import the necessary tools to plot solutions of our equation:

[1]:
```
from matplotlib.pyplot import cm
import matplotlib.pyplot as plt
import numpy as np
import math
```

The numerical values of the constants below yield geophysically reasonable answers and they put the system in the purely oscillatory case, which we shall refer to as the 'critical case', discussed above

[2]:
```
g = 3/4 #gamma
b = 5/3
e = 0.1 #epsilon
r = 1/4
a = 1/8 #alpha
```

3.1 The linear system.

The following code segment sets up a meshgrid (i.e. small rectangular chunks that we can solve the equation in) The bounds in np.arrange describe the size of the meshgrid, or the domain over which we're solving.

[3]:
```
nx, ny = .1, .1
x = np.arange(-2, 2, nx)
y = np.arange(-2, 2, ny)
X, Y = np.meshgrid(x, y)
```

First of all, enter the linearized system below:

[4]:
```
dx =
dy =
```

The block below is a representation of the slope field (but with 'connected' trajectories) on the meshgrid for the linear system. Use the information provided here to run the command plt.streamplot.

[5]:
```
plt.streamplot(X,Y,dx, dy, density=2, cmap= 'jet', arrowsize=1)
```

3.2 Solution of the linear system.

The code block below overlays the analytical solution of the linear system on the slope field obtained above. Modify c_1 and c_2 so that the range of your solution in x is approximately between -1 and 1. Because you are not required to start at a specific initial condition, you may set one of your constants to zero and only modify the other constant. Keep the parameters constant at this point onwards.

[7]:
```
#Note: use np.cos(t) for cos(t), np.sin(t) for sin(t), and np.sqrt() to sqrt()
# use x**2 to enter x^2
g = 3/4 #gamma
b = 5/3
e = 0.1 #epsilon
r = 1/4
a = 1/8 #alpha

c_1 =
c_2 =

w = #you can define omega here in terms of the other parameters if you'd like, it may␣
 ↪make it easier to enter your solution

t=np.linspace(0,50,100)

def x(t):
    return #add your function before this comment

def y(t):
    return #add your function before this comment

plt.streamplot(X,Y,dx, dy, density=2, cmap= 'jet', arrowsize=1)
plt.plot(x(t),y(t))
```

3.3 Exploring the non-critical solution

We now explore the situation where the solutions of the linear system are not purely oscillatory. For this, slightly increase the value of γ to $\gamma = \frac{35}{40}$ and plot again the slope field.

The second part of the block below returns a particular solution, so you should check that it fits with the slope field you obtained. How do successive El Nino events behave?

[14]:
```
g =

plt.xlim(-2, 2)
plt.ylim(-2, 2)

# add code to insert the streamplot here

# the following code plots one particular solution with initial condition x=0.01,y=0
from scipy.integrate import solve_ivp
def thc(t,z,a,b,e,g,r):
    x,y=z
    return [-x + g*(b*x + y), -r*y - a*b*x]

t_eval = np.linspace(0,100,500)

sol_spiral = solve_ivp(thc, [0, 100], [0.01, 0], t_eval=t_eval, args=(a,b,e,g,r),␣
 ↪max_step=0.01)

#plot numerical solution:

plt.plot(sol_spiral.y.T[:, 0], sol_spiral.y.T[:, 1])
```

3.4 The non-linear system

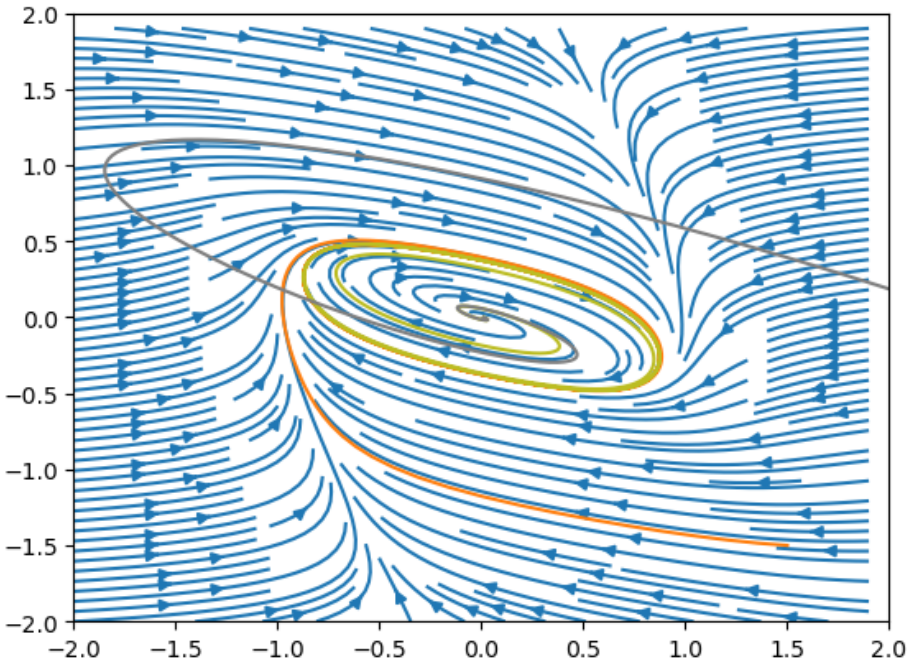
The code below plots the slope field for the full non-linear system and overlays the exponentially growing linearized solution, in grey. It also plots two trajectories of the numerical solution to the non-linear equation so that you can the solutions of the non-linear system and its linearization.

What you see is the existence of a stable **limit cycle** in the non-linear system, this means that all trajectories approach the cycle that is approximated by your linearized solution.

This limit cycle of the non-linear system is in stark contrast to the exponential growth of the solution to the linear equation: This is an example of the **stabilization** effect of the non-linearity.

[16]:
```
#First, find numerical solution to full non-linear system

from scipy.integrate import solve_ivp
def thc(t,z,a,b,e,g,r):
    x,y=z
    return [-x + g*(b*x + y) - e*(b*x + y)**3, -r*y - a*b*x]

t_eval = np.linspace(0,100,500)

# Here the [1.5, -1.5] and [0.01, 0] specify the initial conditions. Feel free to␣
 ↪change them
solout = solve_ivp(thc, [0, 100], [1.5, -1.5], t_eval=t_eval, args=(a,b,e,g,r),␣
 ↪max_step=0.01)
solin = solve_ivp(thc, [0, 100], [0.01, 0], t_eval=t_eval, args=(a,b,e,g,r),␣
 ↪max_step=0.01)

plt.xlim(-2, 2)
plt.ylim(-2, 2)

#plot slopefield:
dx = -X + g*(b*X + Y) - e*(b*X + Y)**3
dy = -r*Y - a*b*X
plt.streamplot(X,Y,dx, dy, density=2, cmap= 'jet', arrowsize=1)

#plot numerical solutions:
plt.plot(solout.y.T[:, 0], solout.y.T[:, 1],'tab:orange')
plt.plot(solin.y.T[:, 0], solin.y.T[:, 1],'tab:olive')

plt.plot(sol_spiral.y.T[:, 0], sol_spiral.y.T[:, 1], 'tab:grey')
```

[16]:
```
[<matplotlib.lines.Line2D at 0x7f20298846d0>]
```



If you feel overwhelmed by the climate emergency, support can be found here:
https://climateemergency.ubc.ca/stem-and-climate-wellbeing-toolkit/