Algorítmica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Programación Orientada a Objetos

Temas para esta sesión

- **Ejercicio**

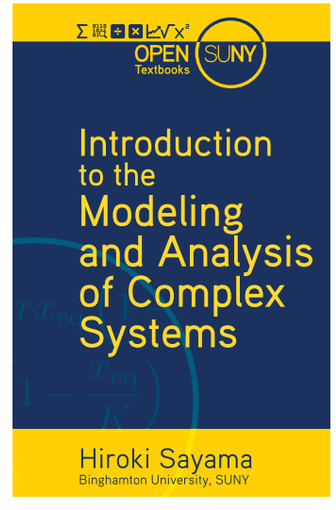
Idea general

Three essential components of computer simulation

Initialize. You will need to set up the initial values for all the state variables of the system.

Observe. You will need to define how you are going to monitor the state of the system. This could be done by just printing out some variables, collecting measurements in a list structure, or visualizing the state of the system.

Update. You will need to define how you are going to update the values of those state variables in every time step. This part will be defined as a function, and it will be executed repeatedly.



Objetivo principal

- Simular el sistema discreto para 40 pasos
- Valores dados por el usuario: **a** y x0

$$x_t = ax_{t-1}$$

```python
class sistemaD:
    __a = 0
    __x = 0

    def __init__(self,val1,val2):
        self.__a = val1
        self.__x = val2

    def observa(self):
        return self.__x

    def actualiza(self):
        self.__x = self.__a * self.__x

# Inicialización
sistema1 = sistemaD(1.01,1)

for x in range(1,41):
    # Observación
    print(sistema1.observa())
    # Actualización
    sistema1.actualiza()
```

Plot

```
# Inicialización
sistema1 = sistemaD(1.01,0.5)

Valores = []

for x in range(1,41):
    # Observación
    Valores.append(sistema1.observa())
    # Actualización
    sistema1.actualiza()

arrayV = np.asarray(Valores)
plt.plot(arrayV)
plt.show()
```

Sistema dinámico con dos variables

- **Ejercicio**

$$x_t = 0.5x_{t-1} + y_{t-1}$$

$$y_t = -0.5x_{t-1} + y_{t-1}$$

$$x_0 = 1, \quad y_0 = 1$$

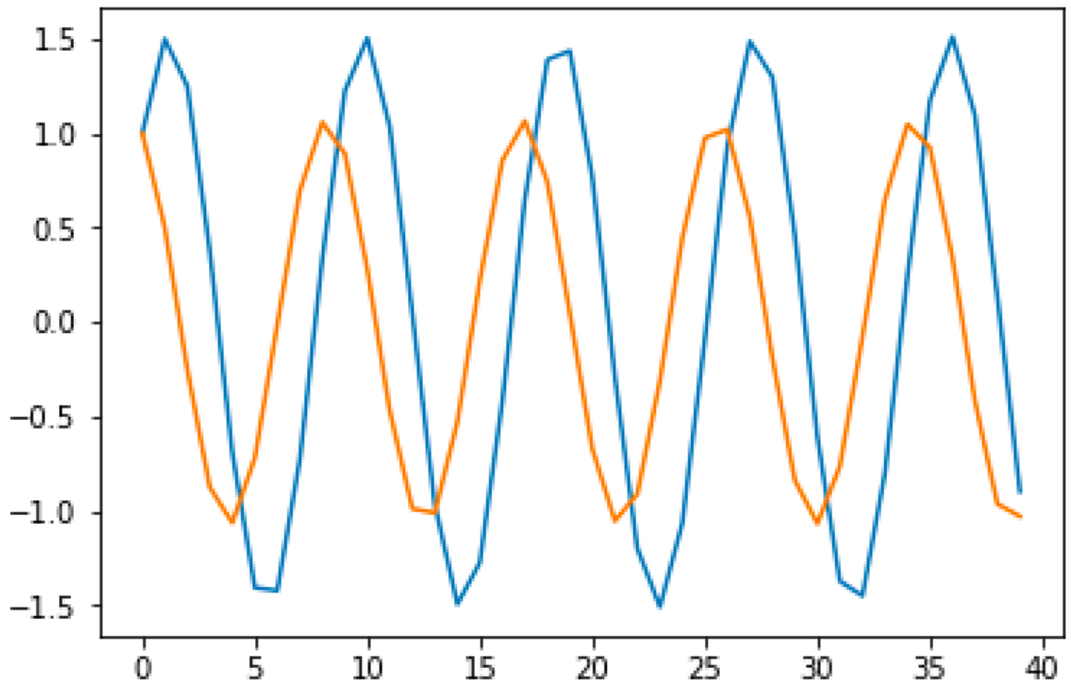
Sistema dinámico con dos variables

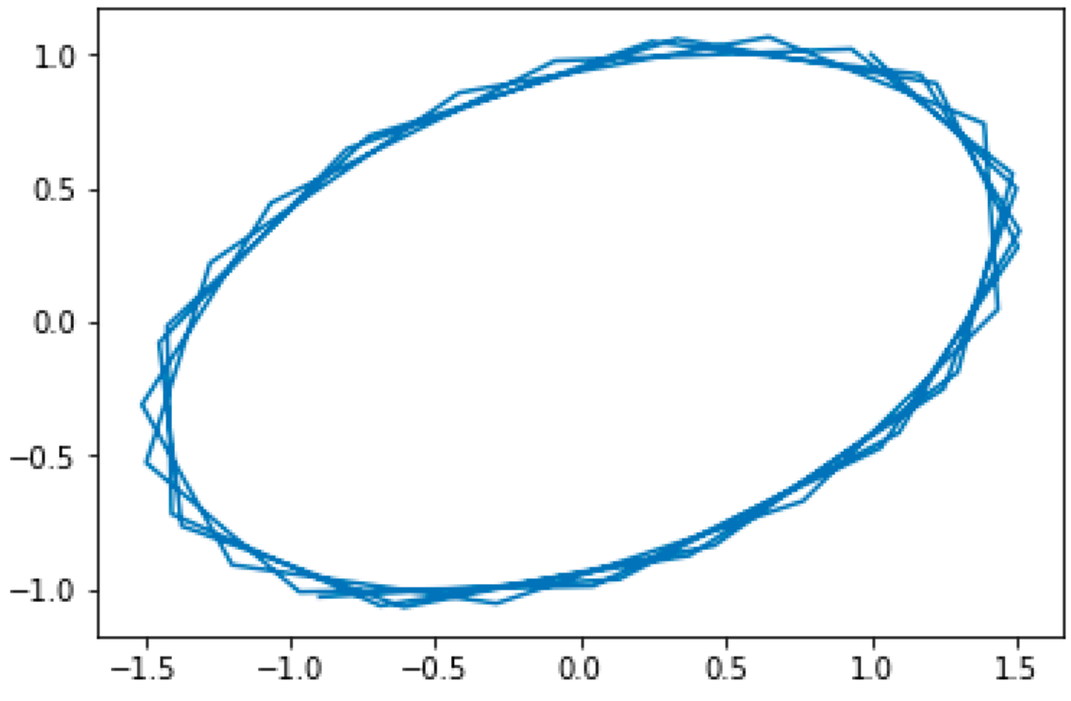
```python
import numpy as np
import matplotlib.pyplot as plt

class sistema2Variables:
    __x = 0
    __y = 0

    def __init__(self,val1,val2):
        self.__x = val1
        self.__y = val2

    def observa(self):
        return self.__x, self.__y

    def actualiza(self):
        nextX =  0.5 * self.__x + self.__y
        nextY = -0.5 * self.__x + self.__y
        self.__x, self.__y  = nextX, nextY
```

Sistema dinámico con dos variables

```
# Inicialización
sistema2 = sistema2Variables(1,1)
Valores  = []

for i in range(1,41):
    # Observación
    # print(sistema1.observa())
    x, y = sistema2.observa()
    Valores.append([x,y])
    # Actualización
    sistema2.actualiza()

### Impresión Final
varArray = np.asarray(Valores)
plt.plot(varArray[:,0])
plt.plot(varArray[:,1])
plt.show()

### Fase
plt.plot(varArray[:,0],varArray[:,1])
plt.show()
```

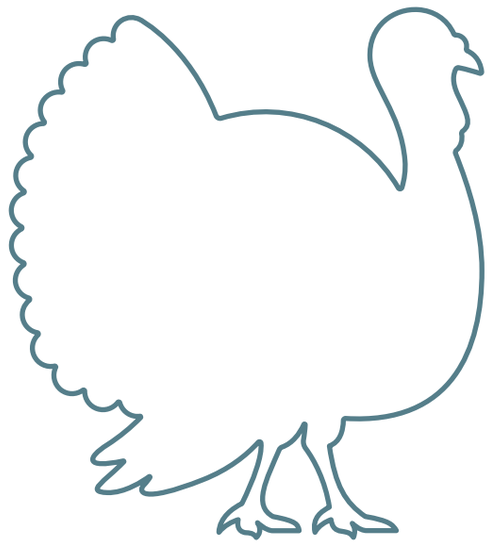
Sistema dinámico con dos variables

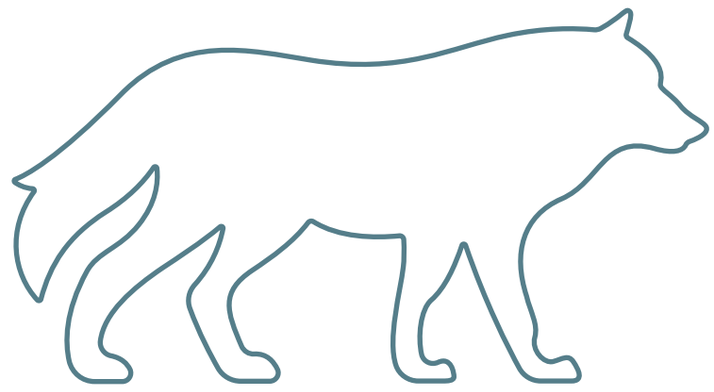




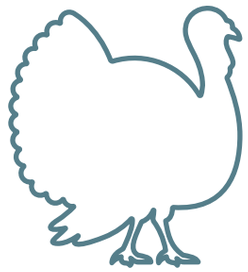
Sistema dinámico con dos variables

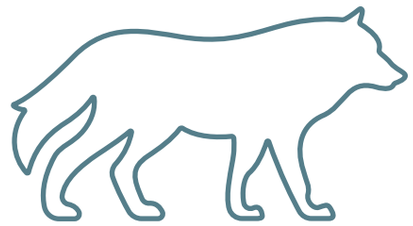
- **Ejercicio**

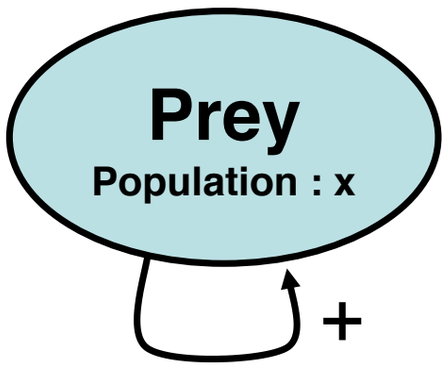




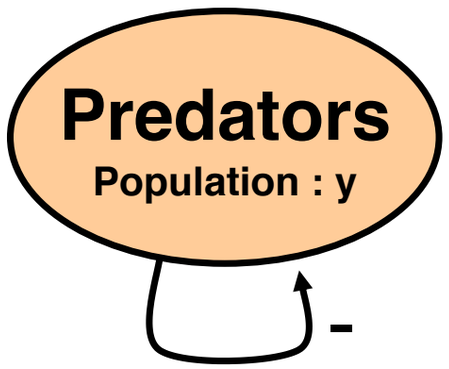
Sistema dinámico con dos variables







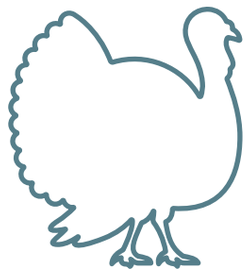
Naturally grows
to carrying capacity
if isolated

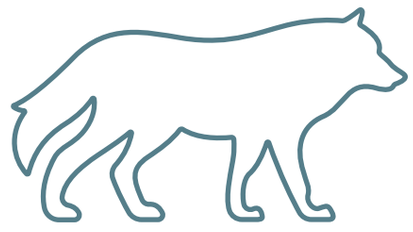


Naturally decays
if isolated

- **Prey grows if there are no predators**
- **Predators die if there are no prey**

Sistema dinámico con dos variables





$$x_t = x_{t-1} + r x_{t-1}$$

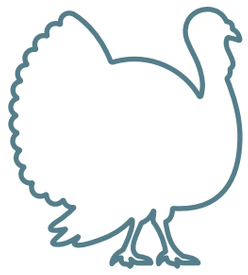
$$y_t = y_{t-1} - d_y y_{t-1}$$

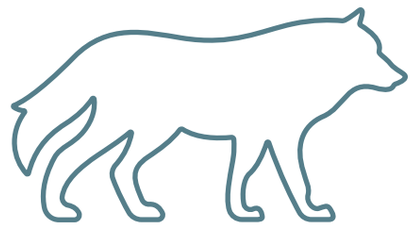
Crecimiento de la población

Decrecimiento de la población

- **Prey grows if there are no predators**
- **Predators die if there are no prey**

Sistema dinámico con dos variables





$$x_t = x_{t-1} + r x_{t-1}\left(1 - \frac{x_{t-1}}{K}\right)$$

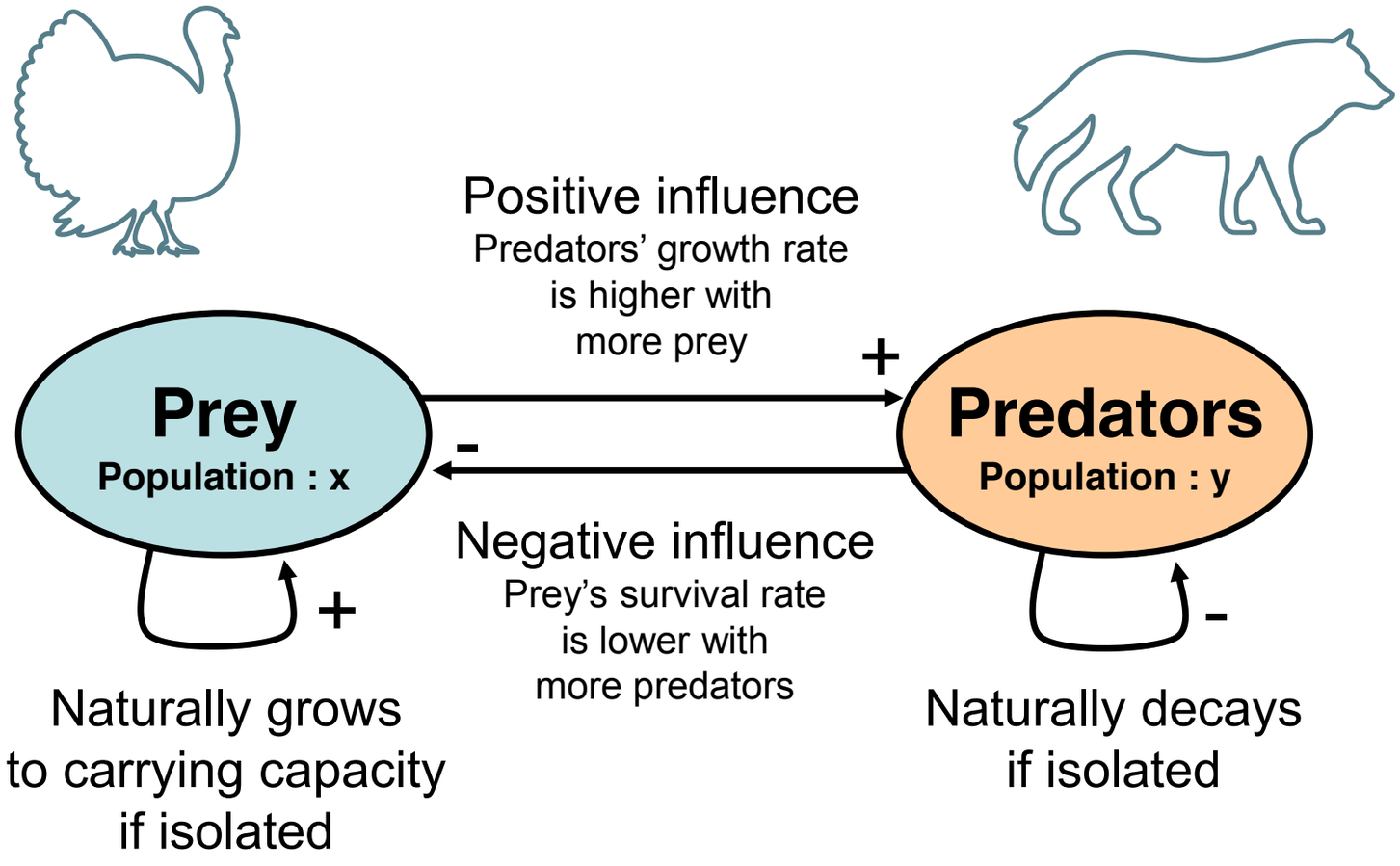
$$y_t = y_{t-1} - d_y y_{t-1}$$

Crecimiento de la población

Decrecimiento de la población

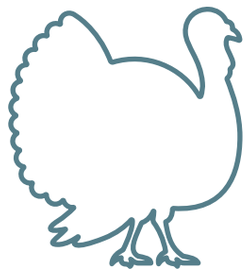
- **Prey grows if there are no predators**
- **Predators die if there are no prey**

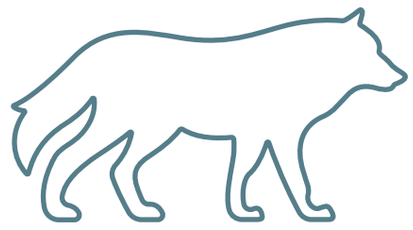
Sistema dinámico con dos variables



- **The prey's death rate increases as the predator population increases**
- **The predators' growth rate increases as the prey population increases**

Sistema dinámico con dos variables





$$x_t = x_{t-1} + r_x x_{t-1}(1 - x_{t-1}/K) \qquad y_t = y_{t-1} - d_y y_{t-1}$$

Crecimiento de la población

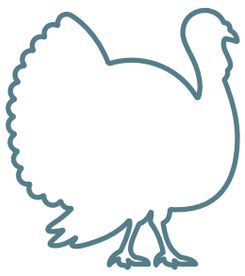
Decrecimiento de la población

$$x_t = x_{t-1} + r x_{t-1}(1 - x_{t-1}/K) - d_x(y_{t-1})x_{t-1} \qquad y_t = y_{t-1} - d y_{t-1} + r_y(x_{t-1})y_{t-1}$$

Decrecimiento de la población

Crecimiento de la población

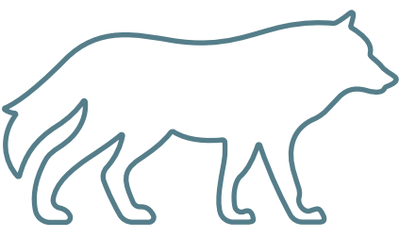
Sistema dinámico con dos variables



$$x_t = x_{t-1} + r x_{t-1}\left(1 - \frac{x_{t-1}}{K}\right) - \left(1 - \frac{1}{b y_{t-1} + 1}\right) x_{t-1}$$

Crecimiento de la población

Decrecimiento de la población



$$y_t = y_{t-1} - d y_{t-1} + c x_{t-1} y_{t-1}$$

Decrecimiento de la población

Crecimiento de la población

Sistema dinámico con dos variables

```python
import numpy as np
import matplotlib.pyplot as plt

class sistemaPandP():
    __r = 0
    __b = 0
    __d = 0
    __c = 0
    __k = 0

    def __init__(self,v1,v2,v3,v4,v5,v6,v7):
        self.__x = v1
        self.__y = v2
        self.__r = v3
        self.__b = v4
        self.__d = v5
        self.__c = v6
        self.__k = v7

    def observa(self):
        return self.__x, self.__y

    def actualiza(self):
        # Actualizar con ecucaciones del sistema prey and predator
        primero = self.__x
        segundo = self.__r * self.__x * (1 - (self.__x / self.__k))
        tercero = (1 - (1 / ((self.__b * self.__y) + 1)) ) * self.__x

        nextX =  primero + segundo - tercero

        nextY =  self.__y - (self.__d * self.__y) + (self.__c * self.__x * self.__y)

        self.__x, self.__y  = nextX, nextY
```

Sistema dinámico con dos variables

```
# Inicialización
sistema3 = sistemaPandP(1,1,1,1,1,1,5)
Valores  = []

for i in range(1,101):
    # Observación
    # print(sistema1.observa())
    x, y = sistema3.observa()
    Valores.append([x,y])
    # Actualización
    sistema3.actualiza()

### Impresión Final
varArray = np.asarray(Valores)
plt.plot(varArray[:,0])
plt.plot(varArray[:,1])
plt.show()

### Fase
plt.plot(varArray[:,0],varArray[:,1])
plt.show()
```

ITAM

Sistema dinámico con dos variables