Trabajo Practico N°1 – Modelos lineales y no lineales

Enunciados

1- Elegir uno de los modelos presentados en el capítulo 1 (sistema masa-resorte-amortiguador o depredador-presa).

2- Realizar un programa interactivo, del caso seleccionado anteriormente, que permita al usuario introducir algunas variables y parámetros de entrada iniciales. El objetivo es estudiar su comportamiento bajo distintas condiciones.

3- Graficar en pantalla los diagramas correspondientes que resulten de la simulación.

- Si sistema es el modelo Depredador-Presa, considere los parámetros de natalidad de liebres y mortalidad de zorros analizados en el caso para una población ideal de 500 liebres y 10 zorros (ver apuntes de cátedra).

4- Deberá presentar un informe que contenga:

- Instrucciones para el uso básico de la aplicación o programa.
- Una explicación de los algoritmos principales utilizados en el programa.
- Pruebas de simulación:
    - Una simulación con los valores por defecto (los vistos en los casos de los apuntes).
    - Dos simulaciones distintas cambiando las condiciones iniciales.

Para cada caso realizar una interpretación de los gráficos, incluyendo capturas de los gráficos.

- Entrega:

La entrega deberá cumplir los siguientes requisitos:

- El código deberá ser presentado en un repositorio git, enviado por mail, o entregado en la presente tarea. Queda abierto a las herramientas utilizadas por los alumnos y alumnas.
- El código debe presentarse convenientemente ordenado y deberá contar con los archivos anteriormente solicitados (informe, código, etc.).
- Entrega antes de la fecha estipulada.

Resolución

En este practica se trabajó con el modelo de Depredador-Presa. Para la resolución de este modelo se utilizaron los apuntes de la catedra, concretamente el capítulo 2.4, **Sistema no lineal simple: el caso presa-predador**.

Desarrollo de Menú

Se realizo el desarrollo de un Menú interactivo para que el usuario ingrese las opciones de configuración de variables que quiera probar y posteriormente ejecute la simulación:

```python
class Menu():
    def show_options(self):
        '''
        Show the options of the menu
        '''
        while True:
            print(OPTION_1)
            print(OPTION_2)
            option = input(SELECT_OPTION)
            if option == "1":
                print(MENU)
                variable_option = int(input(SELECT_OPTION))
                print(START_SIMULATION)
                self.config.load_variables(variable_option)
                predator_prey = PredatorPrey()
                time_result, preys_result, predators_result = predator_prey.run(
                    dt=self.config.dt,
                    initial_time=self.config.initial_time,
                    initial_preys=self.config.hares,
                    initial_predators=self.config.foxes,
                    preys_birth_rate=self.config.hares_birth_rate,
                    preys_death_rate=self.config.hares_death_rate,
                    predators_birth_rate=self.config.foxes_birth_rate,
                    predators_death_rate=self.config.foxes_death_rate,
                    elapsed_time=self.config.weeks,
                    land_capacity=self.config.land_capacity
                )
                print("Times: ", time_result)
                print("Preys: ", preys_result)
                print("Predators: ", predators_result)
                plotter = GraphicsPlotter()
                plotter.plot_population_variation(time_result, preys_result, predators_result)
                plotter.plot_population_variation_with_both_axes(time_result, preys_result, predators_result)
                plotter.plot_phase_diagram(preys_result, predators_result)
                print(END_SIMULATION)
```

Configuración de Variables de Simulación

```python
# Dictionary with the configuration variables of the simulation
VARIABLE_SETS = {
            1: {
                'hares': 500,
                'foxes': 10,
                'weeks': 500,
                'initial_time': 1,
                'final_time': 500,
                'dt': 1,
                'land_capacity': 1400,
                'hares_birth_rate': 0.08,
                'hares_death_rate': 0.002,
                'foxes_birth_rate': 0.0004,
                'foxes_death_rate': 0.2
            },
            2: {
                'hares': 600,
                'foxes': 20,
                'weeks': 600,
                'initial_time': 1,
                'final_time': 600,
                'dt': 1,
                'land_capacity': 1800,
                'hares_birth_rate': 0.1,
                'hares_death_rate': 0.002,
                'foxes_birth_rate': 0.0008,
                'foxes_death_rate': 0.4
            },
```

Simulación

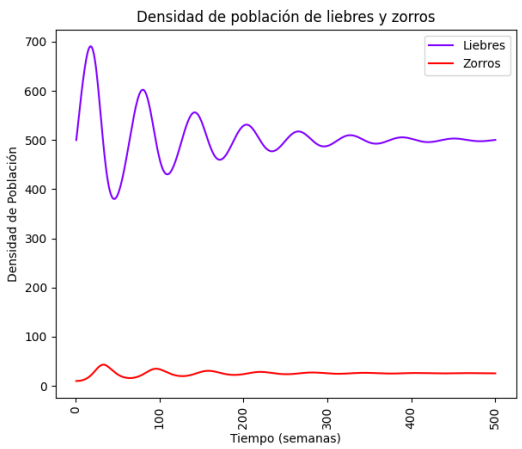
Para la parte de la simulación se trabajó en 3 partes fundamentales, por un lado, la creación de las listas que almacenarían los valores de las poblaciones, siguiendo después con la iteración *for* que actualizaría los valores de dichas poblaciones y finalmente guardando esos valores actualizados en las listas previamente creadas.

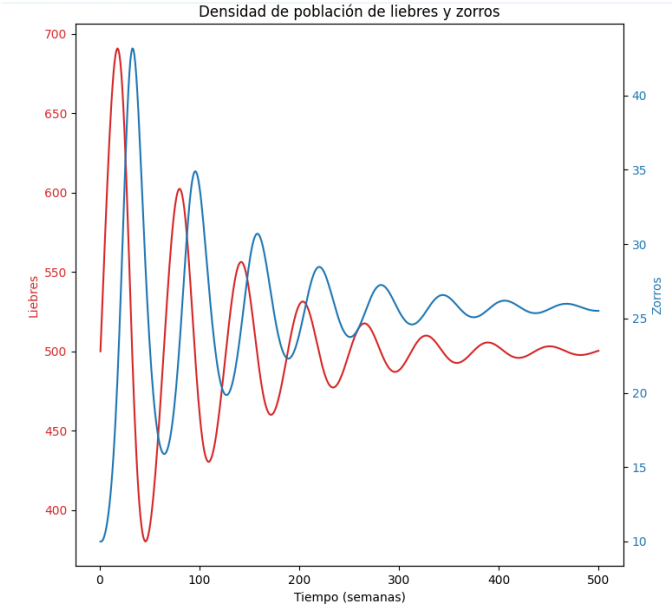
Luego, procedíamos a convertir estas listas en arrays para que nos sean más útiles a la hora de trabajarlos en la parte grafica.

```python
def run(self, dt, initial_time, initial_preys, initial_predators, preys_birth_rate, preys_death_rate,
        predators_birth_rate, predators_death_rate, elapsed_time, land_capacity):
    '''
    Execute the simulation of the predator-prey model
    '''

    preys_list = [initial_preys]
    predators_list = [initial_predators]
    time_list = [initial_time]

    time = initial_time

    for i in range(initial_time, elapsed_time):
        time = time + dt
        hunt_encounter_value = self.hunt_event(preys_list[-1], predators_list[-1])
        actual_capacity = self.get_actual_capacity(preys_list[-1], land_capacity)
        preys_updated = self.get_variations_preys(preys_list[-1], dt, land_capacity, actual_capacity,
                                                  preys_birth_rate, hunt_encounter_value, preys_death_rate)
        predators_updated = self.get_variations_predators(predators_list[-1], dt, predators_death_rate,
                                                          hunt_encounter_value, predators_birth_rate)
        preys_list.append(preys_updated)
        predators_list.append(predators_updated)
        time_list.append(time)

    return np.array(time_list), np.array(preys_list), np.array(predators_list)
```

Gráficos

Para la parte grafica se realizaron concretamente 3 gráficos, por un lado, 2 gráficos que muestran la variación de la población, uno con dos ejes simples y el otro con un eje combinado de las 2 variables presa-depredador.





Además, el tercer grafico es un diagrama de fases que representa los diferentes estados de ambas poblaciones.