# Escuela Politécnica Nacional

*Facultad de Ingeniería de Sistemas*


## *Método Numéricos*

## Proyecto IIB

Integrantes:

Erick Carcelén

Kevin Alvear

Luis Morocho

Andrés Pérez

Profesor: Jonathan Alejandro Zea G.

Fecha de Entrega: 16/08/2024

## *Resumen*

Este informe presenta un análisis detallado del modelo logístico de crecimiento poblacional, implementado a través de una aplicación interactiva desarrollada en Python. El estudio examina la dinámica de poblaciones bajo diferentes condiciones, explorando conceptos clave como puntos fijos, bifurcaciones y comportamiento caótico.

Puntos destacados:

- Se desarrolló una interfaz gráfica que permite a los usuarios experimentar con parámetros clave del modelo logístico.
- El análisis revela cómo pequeños cambios en la tasa de crecimiento pueden llevar a comportamientos drásticamente diferentes, desde la estabilidad hasta el caos.
- Se identificaron y clasificaron diferentes regímenes de comportamiento poblacional, incluyendo extinción, equilibrio estable, oscilaciones periódicas y caos.
- La visualización del diagrama de bifurcación proporciona una comprensión intuitiva de las transiciones entre estos regímenes.
- Se discuten las implicaciones prácticas de estos hallazgos en campos como la ecología y la gestión de recursos naturales.

Este proyecto demuestra la potencia de los modelos matemáticos simples para revelar comportamientos complejos en sistemas naturales, y ofrece una herramienta educativa valiosa para la exploración de conceptos en dinámica no lineal.

## *Metodología*

Para este estudio, se adoptó un enfoque que combina el análisis matemático con la simulación computacional y la visualización de datos. La metodología se dividió en las siguientes fases:

1. Formulación matemática:
   - Se partió de la ecuación logística discreta: $P(n+1) = f(P(n))(1 - P(n))$
   - Se realizó un análisis de puntos fijos y estabilidad para diferentes valores del parámetro f.
2. Implementación computacional:
   - Se desarrolló un programa en Python utilizando las bibliotecas numpy para cálculos numéricos y matplotlib para visualización.
   - Se implementó la ecuación logística como una función iterativa.
3. Desarrollo de la interfaz gráfica:
   - Se creó una interfaz de usuario interactiva utilizando la biblioteca Tkinter de Python.
   - Se incluyeron controles para ajustar parámetros clave como la tasa de crecimiento y la población inicial.
4. Simulaciones y análisis:
   - Se realizaron simulaciones para una amplia gama de valores de f ($0 \leq f \leq 4$).
   - Se generaron series temporales de población y diagramas de bifurcación.
   - Se identificaron y clasificaron diferentes regímenes de comportamiento (extinción, equilibrio, oscilaciones, caos).
5. Visualización de resultados:
   - Se crearon gráficos dinámicos que muestran la evolución temporal de la población.
   - Se generó un diagrama de bifurcación interactivo para visualizar transiciones entre diferentes regímenes.
6. Validación y pruebas:
   - Se verificó la precisión de los cálculos comparando con resultados analíticos conocidos para casos simples.
   - Se realizaron pruebas de usuario para asegurar la usabilidad de la interfaz gráfica.

Esta metodología permitió un análisis integral del modelo logístico, combinando rigor matemático con herramientas computacionales modernas para proporcionar insights valiosos sobre la dinámica de poblaciones.

# *Índice*

# *1 OBJETIVOS*

- Explicar la formulación matemática del modelo logístico de crecimiento poblacional y sus componentes clave.
- Describir los conceptos de puntos fijos y su relevancia en la estabilidad de sistemas dinámicos.
- Investigar el fenómeno de las bifurcaciones en sistemas dinámicos y su manifestación en el modelo logístico de población.
- Determinar las condiciones bajo las cuales la población converge a un punto fijo estable y explicar el significado ecológico de este comportamiento.
- Explorar cómo la variación en el parámetro f afecta la dinámica poblacional, desde la extinción hasta el caos.
- Identificar y clasificar los diferentes tipos de bifurcaciones que ocurren en el modelo logístico a medida que f aumenta.
- Examinar el comportamiento caótico en el modelo logístico y sus implicaciones para la predictibilidad del sistema a largo plazo.
- Elaborar un diagrama de bifurcación que visualice las transiciones dinámicas del sistema a medida que se incrementa f.
- Investigar las aplicaciones prácticas del modelo logístico en la ecología y otras disciplinas, como la economía y la medicina.
- Discutir las implicaciones de los resultados del modelo logístico para la conservación de especies y la gestión de recursos naturales.
- Evaluar la sensibilidad del sistema a las condiciones iniciales y su impacto en la evolución de la población en regímenes caóticos.
- Proponer líneas futuras de investigación que aborden las limitaciones del modelo logístico y exploren modelos más complejos para la dinámica poblacional.

# *2 INTRODUCCIÓN*

El comportamiento de las poblaciones biológicas es un campo de estudio fundamental en la ecología matemática y la teoría de sistemas dinámicos. Desde hace décadas, los científicos han desarrollado diversos modelos matemáticos para comprender cómo varían las poblaciones a lo largo del tiempo bajo diferentes condiciones ambientales y biológicas. Uno de los modelos más emblemáticos y estudiados es el modelo logístico, que ofrece una forma simplificada pero poderosa de representar la dinámica poblacional. Este modelo, formulado a través de una ecuación de recurrencia no lineal, se expresa como $P(n+1) = f * P_n * (1 - P_n)$, donde $P_n$ representa la proporción de la población existente en un tiempo determinado respecto a su capacidad máxima, y $f$ es un parámetro crítico que determina la tasa de crecimiento de la población. Aunque su formulación es relativamente sencilla, el modelo logístico captura una amplia gama de comportamientos dinámicos, desde la extinción de la población hasta patrones complejos y caóticos, dependiendo del valor de $f$.

El interés en el modelo logístico no solo radica en su capacidad para describir diferentes trayectorias poblacionales, sino también en su relevancia para la teoría de sistemas dinámicos y la ecología matemática. A medida que el parámetro de crecimiento $f$ se ajusta, el sistema puede exhibir puntos fijos estables, oscilaciones periódicas, y comportamientos caóticos, todos los cuales tienen implicaciones profundas para la estabilidad y la predictibilidad de las poblaciones en entornos naturales. Estos patrones de comportamiento no solo son teóricamente fascinantes, sino que también ofrecen insights valiosos para la gestión de poblaciones de especies, la conservación de la biodiversidad, y la comprensión de los límites de crecimiento en ecosistemas finitos. A través del análisis del modelo logístico, se pueden identificar las condiciones bajo las cuales las poblaciones pueden prosperar, mantenerse estables, o colapsar, proporcionando un marco conceptual para la toma de decisiones en ecología y biología de la conservación.

En este informe, se profundizará en el análisis de la dinámica poblacional mediante el modelo logístico, enfocándose en cómo los diferentes valores del parámetro $f$ influyen en el comportamiento a largo plazo de la población. Se explorarán conceptos avanzados como los puntos fijos, que representan estados de equilibrio donde la población deja de cambiar, y las bifurcaciones, que son transiciones críticas donde el comportamiento del sistema cambia cualitativamente. Además, se investigará el fenómeno del caos, un estado en el que la población puede exhibir una sensibilidad extrema a las condiciones iniciales, lo que hace que las predicciones a largo plazo sean inherentemente inciertas. Este análisis no solo proporcionará una comprensión más profunda del modelo logístico en sí, sino que también permitirá vislumbrar las implicaciones más amplias de estos conceptos en la ecología y en la gestión sostenible de recursos naturales.

# *3 DESARROLLO*

## 3.1 Descripción del proyecto

La población de una comunidad se puede modelar con la fórmula:

$$P_{n+1} = f * P_n * (1 - P_n)$$

Donde:

- $P_n \in [0,1]$ representa el porcentaje de la población existente en el año n con respecto al máximo,
- $f \in [0,4]$ es la constante de fertilidad y representa la tasa de crecimiento de la población.

Comportamiento:

Como se verá a continuación, la población presenta un comportamiento diferente dependiendo de la constante de fertilidad f.

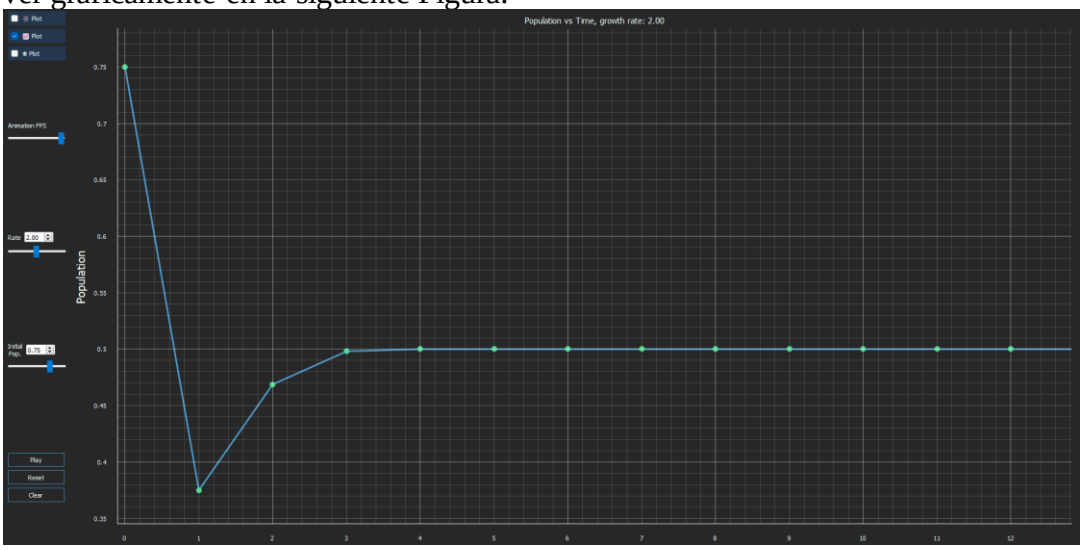
- Cuando $0 \leq f \leq 1$ la población eventualmente muere independiente del valor de población inicial $P_0$.
    - En este caso se dice que tiene 0 bifurcación.
- Cuando $1 < f < 2$ la población se estabiliza independiente del valor de población inicial $P_0$.
    - En este caso se dice que tiene 1 bifurcación.
- Para ciertos valores de $f$, la población oscila entre dos valores.
    - En este caso se dice que tiene 2 bifurcaciones.
- Para los valores restantes de $f$, debe determinar el comportamiento de la población y el número de bifurcaciones.

Ejemplos:

❖ Caso 1:
Dado $f = 2$, $P_0 = 0.75$

$P_1 = 2 * 0.75 * (1 - 0.75) = 0.375$
$P_2 = 2 * 0.375 * (1 - 0.375) = 0.46875$
$P_3 = 2 * 0.46875 * (1 - 0.46875) = \frac{255}{512} \approx 0.49805$
$P_4 = 2 * \frac{255}{512} * \left(1 - \frac{255}{512}\right) \approx 0.4999\ldots$
$P_5 = 2 * 0.4999\ldots * (1 - 0.4999\ldots) = 0.4999\ldots$
$P_6 = 2 * 0.4999\ldots * (1 - 0.4999\ldots) = 0.4999\ldots$

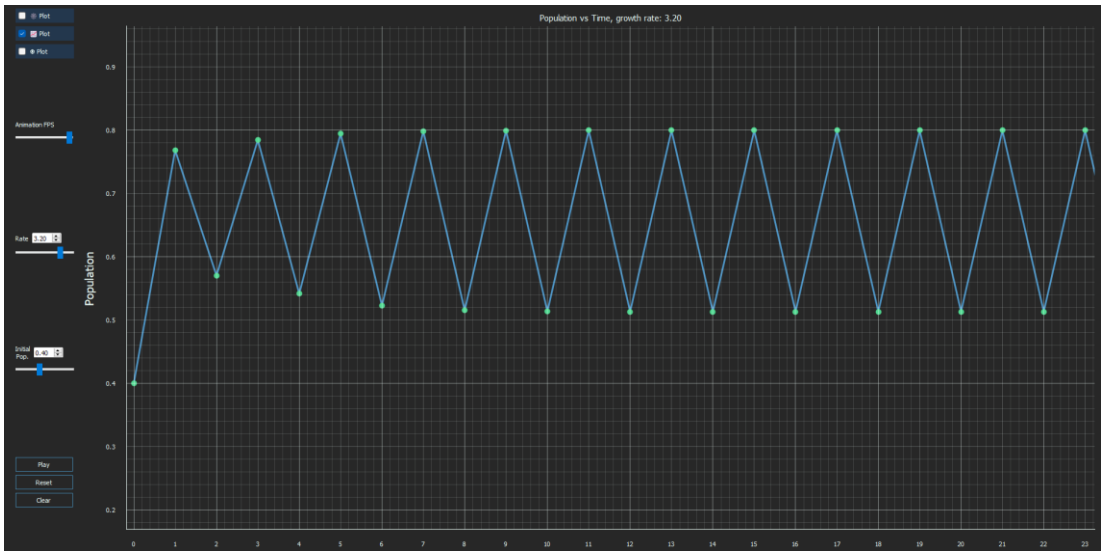
En este ejemplo, la población converge al valor de 0.5 (1 bifurcación), esto se puede ver gráficamente en la siguiente Figura:




❖ Caso 2:
Dado $f = 3.2$, $P_0 = 0.4$

| $P_n$ | Valor |
|---|---|
| $P_0$ | 0.4 |
| $P_1$ | 0.768 |
| $P_2$ | 0.5702 |
| $P_3$ | 0.7842 |
| $P_4$ | 0.5415 |
| $P_5$ | 0.7945 |
| $P_6$ | 0.5225 |
| $P_7$ | 0.7984 |
| $P_8$ | 0.5151 |
| $P_9$ | 0.7993 |
| $P_{10}$ | 0.5134 |
| $P_{11}$ | 0.7994 |
| $P_{12}$ | 0.5131 |
| $P_{13}$ | 0.7995 |
| $P_{14}$ | 0.5131 |

En este caso, vemos que la población no converge a ningún valor, sino que oscila entre 0.7995 y 0.5130 (2 bifurcaciones), esto se puede ver gráficamente en la siguiente Figura:




## 3.2 Puntos Fijos e Iteración Funcional

Un punto fijo de una función es un valor que permanece constante bajo la aplicación repetida de la función. En otras palabras, si una población alcanza un punto fijo bajo la iteración de la función, continuará en ese nivel indefinidamente, siempre y cuando no haya perturbaciones externas que alteren el sistema. Para el modelo logístico dado, los puntos fijos $P^*$ se encuentran resolviendo la ecuación:

$$P^* = f * P^* * (1 - P^*)$$

Esta ecuación cuadrática puede ser simplificada para encontrar dos soluciones potenciales: $P^* = 0$ y $P^* = \frac{f-1}{f}$. El primer punto fijo, $P^* = 0$, representa la extinción de la población, un escenario que puede ocurrir cuando la tasa de crecimiento $f$ es baja, insuficiente para sostener a la población a lo largo del tiempo.

El segundo punto fijo, $P^* = \frac{f-1}{f}$, es de particular interés porque su estabilidad depende del valor de $f$. La estabilidad de un punto fijo se analiza utilizando la derivada de la función iterada en ese punto. Si la magnitud de la derivada $|f'(P^*)|$ es menor que 1, el punto fijo es estable, lo que significa que las pequeñas desviaciones de la población respecto a este valor tenderán a corregirse con el tiempo, permitiendo que la población converja a $P^*$. Si $|f'(P^*)|$ es mayor que 1, el punto fijo es inestable, lo que significa que cualquier desviación de este valor hará que la población diverja, llevando a un comportamiento oscilatorio o incluso caótico.

## 3.3 Comportamiento de la Población para $1 \leq f \leq 2$

En el intervalo donde $f$ está entre 1 y 2, el comportamiento de la población es relativamente sencillo y predecible. El punto fijo no trivial $P^* = \frac{f-1}{f}$ es estable, lo que implica que la población, independientemente de su valor inicial $P_0$, tenderá a estabilizarse en este punto con el tiempo. Esta situación es característica de un sistema dinámico en equilibrio, donde la tasa de crecimiento es suficiente para sostener la población, pero no lo suficientemente alta como para inducir oscilaciones.

Por ejemplo, cuando $f = 2$, el punto fijo se calcula como $P^* = \frac{2-1}{2} = 0.5$. Como se observó en el "Caso 1", la población comienza en un valor inicial, y tras algunas iteraciones, converge rápidamente a 0.5, mostrando que el sistema ha alcanzado un equilibrio estable. Este comportamiento ilustra cómo el sistema encuentra un balance entre la tasa de crecimiento y la capacidad de carga ambiental, evitando tanto la extinción como el crecimiento ilimitado.

## 3.4 Bifurcaciones y Comportamiento para $f \geq 0$

El concepto de bifurcación es crucial para entender cómo cambios en el parámetro $f$ pueden alterar radicalmente el comportamiento del sistema. Una bifurcación ocurre cuando un cambio en un parámetro provoca un cambio cualitativo en la dinámica del sistema, como la transición de un punto fijo estable a un ciclo periódico o a un comportamiento caótico.

- 0 bifurcación ($0 \leq f \leq 1$): En este rango, el único punto fijo estable es $P^* = 0$. Esto significa que la población inevitablemente se extinguirá, sin importar el valor inicial de $P_0$. Este comportamiento refleja una situación donde la tasa de crecimiento $f$ es demasiado baja para sostener la población a largo plazo. Este escenario podría ser típico en ambientes con recursos extremadamente limitados o con factores de mortalidad que superan la tasa de natalidad.
- 1 bifurcación ($1 < f < 3$): En este intervalo, el punto fijo no trivial $P^* = \frac{f-1}{f}$ se convierte en el atractor del sistema. Aquí, la población se estabiliza en un único valor, representando un estado de equilibrio. Este comportamiento indica que, aunque la población enfrenta limitaciones, el ambiente aún es capaz de sostener un nivel constante de individuos, sin que la población caiga a cero ni crezca de manera descontrolada.
- 2 bifurcaciones ($3 \leq f \leq 3.45$): A medida que $f$ aumenta más allá de 3, el sistema experimenta una bifurcación de periodo 2. Esto significa que la población ya no converge a un único valor, sino que comienza a oscilar entre dos valores diferentes. Este comportamiento cíclico refleja un sistema en el cual la tasa de crecimiento es lo suficientemente alta para evitar un punto de equilibrio estable, pero no tan alta como para inducir el caos. En el "Caso 2",

con $f = 3.2$, observamos precisamente este fenómeno, donde la población oscila entre dos valores en un ciclo de periodo 2.

- Bifurcaciones adicionales y caos ($f > 3.45$): A medida que $f$ sigue aumentando, el sistema sufre bifurcaciones adicionales, pasando de ciclos de periodo 2 a ciclos de periodos más largos (4, 8, etc.), hasta que eventualmente entra en un régimen caótico. En el caos, la población no sigue un patrón predecible y pequeñas diferencias en las condiciones iniciales pueden llevar a trayectorias completamente diferentes, lo que caracteriza un sistema extremadamente sensible y complejo. Este comportamiento caótico, aunque difícil de predecir, es una manifestación de la sensibilidad a las condiciones iniciales, un concepto clave en la teoría del caos.

## 3.5 Análisis del Comportamiento de la Población para Valores Altos de $f$

Para valores altos de $f$, el comportamiento del sistema se vuelve cada vez más complejo y difícil de predecir. Cuando $f$ supera el umbral de aproximadamente 3.45, el sistema comienza a mostrar señales de comportamiento caótico. En este régimen, la dinámica de la población no sigue un patrón periódico estable y se vuelve extremadamente sensible a las condiciones iniciales. Este tipo de comportamiento es típico en sistemas no lineales complejos y es uno de los pilares de la teoría del caos.

El caos en el modelo logístico puede visualizarse mediante diagramas de bifurcación, que muestran cómo los valores de $P_n$ se comportan a medida que $f$ aumenta. Estos diagramas revelan que, en el caos, no hay un número fijo de bifurcaciones; en cambio, el sistema puede moverse de un ciclo a otro, y eventualmente entrar en un régimen donde la población oscila de manera aparentemente aleatoria.

Este comportamiento caótico tiene implicaciones profundas en la biología y otras ciencias, ya que muestra que incluso sistemas simples pueden producir resultados altamente impredecibles. Este es un recordatorio importante de que, en modelos de sistemas complejos, las predicciones a largo plazo pueden volverse imposibles, ya que pequeñas variaciones en los parámetros o en las condiciones iniciales pueden llevar a resultados drásticamente diferentes.

## 3.6 Aplicaciones Prácticas y Consideraciones en la Ecología y más allá

El análisis de bifurcaciones y el comportamiento caótico en modelos de población no solo es relevante desde una perspectiva teórica, sino que también tiene aplicaciones prácticas significativas en ecología, conservación y gestión de recursos naturales. Comprender cómo las poblaciones pueden entrar en regímenes caóticos o bifurcarse a diferentes estados puede ayudar a los ecólogos a prever situaciones donde las poblaciones podrían volverse inestables, lo que podría ser crucial para la conservación de especies en peligro o la gestión sostenible de recursos.

Además, estos modelos pueden aplicarse más allá de la ecología, en campos como la economía (modelos de oferta y demanda), la ingeniería (sistemas de control), y la medicina (dinámica de enfermedades). En todos estos campos, comprender la dinámica no lineal y el comportamiento caótico es fundamental para el diseño de estrategias robustas que puedan manejar la complejidad y la incertidumbre inherentes a estos sistemas.

## 3.7 Documentación

Este código implementa una aplicación gráfica en Python usando la biblioteca Tkinter, la cual simula y visualiza la dinámica poblacional a través de la ecuación logística y permite al usuario observar la evolución de la población y un diagrama de bifurcación a medida que se ajustan ciertos parámetros.

- Importaciones:

```python
import tkinter as tk
from tkinter import ttk
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
```

  - tkinter: Biblioteca estándar de Python para crear interfaces gráficas de usuario (GUIs).
  - ttk: Submódulo de Tkinter que proporciona widgets temáticos (estilizados) para mejorar la apariencia.
  - numpy: Biblioteca para realizar cálculos numéricos eficientes con matrices y vectores.
  - matplotlib.pyplot: Submódulo de Matplotlib para crear gráficos.
  - FigureCanvasTkAgg: Adaptador de Matplotlib para incrustar gráficos en una aplicación Tkinter.

- Función para la Dinámica Poblacional:

```python
# Función para calcular la dinámica poblacional utilizando la ecuación logística
def logistic_map(rate, initial_pop, steps=100):
    population = np.zeros(steps)
    population[0] = initial_pop
    for i in range(1, steps):
        population[i] = rate * population[i - 1] * (1 - population[i - 1])
    return population
```

  - logistic_map: Esta función calcula la dinámica poblacional utilizando la ecuación logística, que es un modelo simple para describir cómo una

población crece en función de una tasa de crecimiento rate y una capacidad de carga implícita.

- Variables:
  - rate: La tasa de crecimiento de la población.
  - initial_pop: La población inicial.
  - steps: Número de pasos temporales en la simulación.
- Proceso: La población en cada paso se calcula usando la fórmula del modelo logístico: population[i] = rate * population[i-1] * (1 - population[i-1]).

- Función para Graficar la Bifurcación:

```
# Función para actualizar la gráfica de bifurcación
def plot_bifurcation():
    ax.clear()
    ax.set_xlabel('Tasa de Crecimiento (r)', fontsize=12)
    ax.set_ylabel('Población', fontsize=12)
    ax.set_title('Diagrama de Bifurcación', fontsize=14)
    ax.grid(True, which='both', linestyle='--', color='gray', alpha=0.7)

    r_values = np.linspace(0, 4, 1000)
    population = initial_pop_slider.get()
    last_values = 50  # Considerar los últimos valores para visualizar la bifurcación

    for r in r_values:
        pop = logistic_map(r, population, 1000)
        ax.plot([r] * last_values, pop[-last_values:], ',k', alpha=0.25)

    canvas.draw()
```

- plot_bifurcation: Esta función crea un diagrama de bifurcación para visualizar cómo la población cambia a medida que se varía la tasa de crecimiento r.
- Proceso:
  - Limpia la gráfica (ax.clear()), establece los títulos y etiquetas de los ejes, y configura la cuadrícula.
  - r_values genera un array de valores para la tasa de crecimiento desde 0 hasta 4, que se utilizan para calcular la dinámica poblacional.
  - La población inicial se obtiene del valor actual de la slider (initial_pop_slider.get()).
  - Para cada valor de r, se calcula la población durante 1000 pasos y se grafican los últimos 50 valores para mostrar la bifurcación.
  - Finalmente, se actualiza la gráfica con canvas.draw().

- Función para Actualizar los Sliders:

```
# Función para mostrar los valores de los sliders
def update_rate(*args):
    rate_value_label.config(text=f"{rate_slider.get():.2f}")

def update_initial_pop(*args):
    initial_pop_value_label.config(text=f"{initial_pop_slider.get():.2f}")
```

  - update_rate y update_initial_pop: Estas funciones actualizan las etiquetas que muestran el valor actual de las sliders cada vez que estos son ajustados. Los valores se muestran con dos decimales.

- Función para Iniciar la Animación:

```
# Función para iniciar la animación
def start_animation():
    global population, steps
    rate = rate_slider.get()
    initial_pop = initial_pop_slider.get()
    steps = 10

    population = logistic_map(rate, initial_pop, steps)
    animate_plot()
```

  - start_animation: Esta función inicia la simulación de la dinámica poblacional.
    - Obtiene los valores actuales de la tasa de crecimiento y la población inicial desde las sliders.
    - Define steps como el número de pasos temporales a simular.
    - Calcula la dinámica poblacional utilizando logistic_map.
    - Llama a animate_plot para comenzar la animación de la gráfica.

- Función para Animar la Gráfica:

```
# Función para actualizar la gráfica de forma animada
def animate_plot(i=0):
    if i == 0:
        ax.clear()
        ax.set_ylim(0, 1)
        ax.set_xlim(0, steps - 1)
        ax.set_xlabel('Paso de Tiempo', fontsize=12)
        ax.set_ylabel('Población', fontsize=12)
        ax.set_title('Dinámica de la Población', fontsize=14)
        ax.grid(True, which='both', linestyle='--', color='gray', alpha=0.7)

    if i < steps:
        ax.plot(range(i + 1), population[:i + 1], marker='o', color='#007acc', linestyle='-', linewidth=2)
        canvas.draw()
        root.after(200, animate_plot, i + 1)
```

- animate_plot: Controla la animación de la gráfica.
  - Condiciones iniciales: Si i == 0, limpia la gráfica y establece los límites y etiquetas de los ejes.
  - Dibuja la curva: A medida que i aumenta, dibuja la curva de la población desde el paso 0 hasta el actual.
  - Llamadas recursivas: Utiliza root.after(200, animate_plot, i + 1) para actualizar la gráfica cada 200 milisegundos, incrementando i en 1 hasta alcanzar el número total de pasos.

- Funciones para Reiniciar Sliders y Limpiar Gráfica:

```
# Función para reiniciar los valores de los sliders
def reset_sliders():
    rate_slider.set(2.0)
    initial_pop_slider.set(0.75)

# Función para limpiar la gráfica
def clear_plot():
    ax.clear()
    ax.set_ylim(0, 1)
    ax.set_xlim(0, 9)
    canvas.draw()
```

  - reset_sliders: Restablece las sliders a sus valores predeterminados: tasa de crecimiento r = 2.0 y población inicial 0.75.
  - clear_plot: Limpia la gráfica sin realizar ningún cálculo o dibujo adicional.

- Configuración de la Ventana Principal:

```
# Configuración de la ventana principal
root = tk.Tk()
root.title("Dinámica Poblacional")
root.configure(bg="#f4f4f4")  # Fondo claro
```

  - Ventana principal: Configura la ventana principal (root) de la aplicación con un título y un color de fondo.

- Creación de la Gráfica:

```
# Crear la figura y el eje para la gráfica
fig, ax = plt.subplots(figsize=(6, 4), dpi=100)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.get_tk_widget().grid(row=0, column=1, rowspan=6, padx=20, pady=20)
```

  - Gráfica: Crea una figura de Matplotlib (fig) y un eje (ax) para dibujar las gráficas.

- Canvas: El canvas (FigureCanvasTkAgg) permite que la figura se muestre dentro de la ventana Tkinter, colocándola en una cuadrícula.

- Creación del Marco de Controles:

```
# Crear un marco para los controles
control_frame = ttk.Frame(root, padding="10 10 10 10", style="TFrame")
control_frame.grid(row=0, column=0, sticky="nsew", padx=10, pady=10)
```

- control_frame: Es un marco que contiene todos los controles (sliders, botones) y está estilizado con ttk.

- Creación de Controles y Sliders:

```
# Crear controles de la interfaz gráfica
ttk.Label(control_frame, text="Tasa de Crecimiento (r)", style="TLabel").grid(row=0, column=0, sticky="w")

# Slider Rate
rate_slider = ttk.Scale(control_frame, from_=0.0, to=4.0, orient="horizontal")
rate_slider.set(2.0)
rate_slider.grid(row=1, column=0, sticky="ew", pady=5)

# Definición de la etiqueta para mostrar el valor del slider Rate
rate_value_label = ttk.Label(control_frame, text="2.00", style="TLabel")
rate_value_label.grid(row=1, column=1, sticky="w", padx=5)

# Conectar el slider a la función de actualización
rate_slider.bind("<Motion>", update_rate)

ttk.Label(control_frame, text="Población Inicial", style="TLabel").grid(row=2, column=0, sticky="w")

# Slider Initial Pop
initial_pop_slider = ttk.Scale(control_frame, from_=0.0, to=1.0, orient="horizontal")
initial_pop_slider.set(0.75)
initial_pop_slider.grid(row=3, column=0, sticky="ew", pady=5)

# Definición de la etiqueta para mostrar el valor del slider Initial Pop
initial_pop_value_label = ttk.Label(control_frame, text="0.75", style="TLabel")
initial_pop_value_label.grid(row=3, column=1, sticky="w", padx=5)

# Conectar el slider a la función de actualización
initial_pop_slider.bind("<Motion>", update_initial_pop)
```

- Sliders y Etiquetas: Se crean sliders para controlar la tasa de crecimiento y la población inicial, junto con etiquetas para mostrar sus valores actuales.

- Creación de Botones:

```
# Botón para iniciar la animación
play_button = ttk.Button(control_frame, text="Iniciar Animación", command=start_animation, style="TButton")
play_button.grid(row=4, column=0, pady=5, sticky="ew")

# Botón para graficar la bifurcación
bifurcation_button = ttk.Button(control_frame, text="Graficar Bifurcación", command=plot_bifurcation, style="TButton")
bifurcation_button.grid(row=5, column=0, pady=5, sticky="ew")

# Botón para reiniciar los sliders
reset_button = ttk.Button(control_frame, text="Reiniciar", command=reset_sliders, style="TButton")
reset_button.grid(row=6, column=0, pady=5, sticky="ew")

# Botón para limpiar la gráfica
clear_button = ttk.Button(control_frame, text="Limpiar Gráfica", command=clear_plot, style="TButton")
clear_button.grid(row=7, column=0, pady=5, sticky="ew")
```

  - Botones: Botones que permiten al usuario iniciar la animación, graficar la bifurcación, reiniciar las sliders, y limpiar la gráfica.

- Creación de Estilo y Ejecución:

```
# Configurar las columnas para que se ajusten al tamaño del contenido
root.grid_columnconfigure(0, weight=1)
root.grid_columnconfigure(1, weight=3)
root.grid_rowconfigure(0, weight=1)

# Aplicar estilos personalizados a los widgets
style = ttk.Style()
style.configure("TFrame", background="#f4f4f4")
style.configure("TLabel", background="#f4f4f4", font=("Arial", 10))
style.configure("TButton", font=("Arial", 10), padding=6)
style.configure("TScale", background="#f4f4f4")

# Ejecutar la aplicación
root.mainloop()
```

  - Estilo: Aplica estilos personalizados a los diferentes widgets utilizando ttk.Style().
  - Ejecución de la aplicación: Finalmente, root.mainloop() inicia el bucle principal de la aplicación, permitiendo que la interfaz gráfica responda a eventos y acciones del usuario.

# *4 CONCLUSIONES Y RECOMENDACIONES*

## 4.1 Conclusiones

- El código implementa una simulación de la dinámica poblacional utilizando el modelo logístico. Este modelo es un clásico en estudios de poblaciones, utilizado para prever cómo una población crece en función de una tasa de crecimiento y un límite de capacidad.
- La aplicación permite al usuario visualizar tanto la evolución temporal de la población como el diagrama de bifurcación para distintas tasas de crecimiento (r). El diagrama de bifurcación es útil para entender la naturaleza caótica del modelo a medida que la tasa de crecimiento aumenta.
- La interfaz desarrollada con Tkinter proporciona controles interactivos que permiten al usuario ajustar la tasa de crecimiento y la población inicial, así como iniciar animaciones y graficar el diagrama de bifurcación.
- La aplicación es capaz de mostrar de manera visual cómo pequeñas variaciones en los parámetros pueden llevar a comportamientos muy diferentes en el sistema, algo que es típico en sistemas caóticos.
- El diseño de la interfaz es simple y funcional, adecuado para propósitos educativos o experimentales en la visualización de dinámicas no lineales.
- Se utiliza la biblioteca Matplotlib para la generación de gráficos, lo que es apropiado para este tipo de visualizaciones científicas. Sin embargo, no se emplean técnicas avanzadas para optimizar el rendimiento, lo que podría ser un área de mejora si se requiere manejar una mayor cantidad de datos o pasos temporales.

## 4.2 Recomendaciones

- Se puede mejorar la experiencia del usuario proporcionando actualizaciones en tiempo real al mover las sliders, sin necesidad de realizar acciones adicionales como presionar un botón.
- Agregar opciones para personalizar otros aspectos del modelo, como la duración de la simulación o la posibilidad de introducir diferentes funciones de crecimiento.
- El código actual fija el número de pasos temporales en 100 o 1000. Permitir al usuario modificar este parámetro podría ofrecer más flexibilidad y una comprensión más profunda del sistema.
- Para simulaciones largas o con muchas bifurcaciones, se podría optimizar el código, por ejemplo, limitando la cantidad de datos gráficos mostrados al usuario o utilizando técnicas como decimación de datos.
- Considerar la implementación de mejoras en accesibilidad, como atajos de teclado o descripciones auditivas para personas con discapacidades visuales.
- Si se planea expandir esta herramienta para usos más complejos, sería importante considerar cómo escalar el código para manejar más variables o modelos más complejos sin comprometer el rendimiento.

# 5 REFERENCIAS


[1]. MAY, R. M. (1976). SIMPLE MATHEMATICAL MODELS WITH VERY COMPLICATED DYNAMICS. NATURE, 261(5560), 459-467.

[2]. STROGATZ, S. H. (2018). NONLINEAR DYNAMICS AND CHAOS: WITH APPLICATIONS TO PHYSICS, BIOLOGY, CHEMISTRY, AND ENGINEERING. CRC PRESS.

[3]. FEIGENBAUM, M. J. (1978). QUANTITATIVE UNIVERSALITY FOR A CLASS OF NONLINEAR TRANSFORMATIONS. JOURNAL OF STATISTICAL PHYSICS, 19(1), 25-52.

[4]. HASTINGS, A., HOM, C. L., ELLNER, S., TURCHIN, P., & GODFRAY, H. C. J. (1993). CHAOS IN ECOLOGY: IS MOTHER NATURE A STRANGE ATTRACTOR?. ANNUAL REVIEW OF ECOLOGY AND SYSTEMATICS, 24(1), 1-33.

[5]. VERHULST, P. F. (1838). NOTICE SUR LA LOI QUE LA POPULATION SUIT DANS SON ACCROISSEMENT. CORRESP. MATH. PHYS., 10, 113-121.

[6]. DEVANEY, R. L. (2018). AN INTRODUCTION TO CHAOTIC DYNAMICAL SYSTEMS. CRC PRESS.

[7]. PYTHON SOFTWARE FOUNDATION. (2024). PYTHON LANGUAGE REFERENCE, VERSION 3.X. AVAILABLE AT HTTP://WWW.PYTHON.ORG

[8]. HUNTER, J. D. (2007). MATPLOTLIB: A 2D GRAPHICS ENVIRONMENT. COMPUTING IN SCIENCE & ENGINEERING, 9(3), 90-95.

[9]. THE TKINTER TEAM. (2024). TKINTER - PYTHON INTERFACE TO TCL/TK. AVAILABLE AT HTTPS://DOCS.PYTHON.ORG/3/LIBRARY/TKINTER.HTML

[10]. NUMPY COMMUNITY. (2024). NUMPY USER GUIDE. AVAILABLE AT HTTPS://NUMPY.ORG/DOC/