Simulación de un Ecosistema Artificial: Un Modelo Computacional de la Dinámica Depredador - Presa

Lotka-Volterra y Runge-Kutta 4

Diego Sotelo
Alexis Gonzales
Paolo Villavicencio
Alvaro Salazar
Facultad de Ingeniería de Sistemas e Informática

Overleaf FISI ▪ *flowxy.org*

November 18, 2025

Tabla de contenidos

Jacobi Iteration

Definition

The **Jacobi Method** solves the linear system $A\mathbf{x} = \mathbf{b}$ iteratively:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right)$$

It converges if A is strictly diagonally dominant or symmetric positive definite.

Python Implementation — Jacobi

```python
import numpy as np

A = np.array([[10, -1, 2],
              [-1, 11, -1],
              [2, -1, 10]], float)
b = np.array([6, 25, -11], float)

x = np.zeros_like(b)
for k in range(8):
    x_new = np.zeros_like(x)
    for i in range(len(A)):
        s = sum(A[i,j]*x[j] for j in range(len(A)) if j != i)
        x_new[i] = (b[i] - s)/A[i,i]
    x = x_new
    print(f"Iteration {k+1}: {x}")
```

Dinámica de poblaciones con dos especies

Las interacciones entre dos especies pueden ser positivas (+), negativas (-) o neutras (0). Se definen dos tipos específicos de relación: el **neutralismo (0 0)**, donde las poblaciones no se afectan mutuamente, y el **mutualismo (+ +)**, donde ambas se benefician. Este mutualismo puede ser **no obligatorio**, si la relación no es vital para la supervivencia, u **obligatorio**, si es esencial para que ambas poblaciones sobrevivan.

Siendo *P(t)* el tamaño de la población en el instante t, el modelo exponencial presupone que la tasa de aumento de la población es proporcional a la población en ese instante:

$$\frac{dP}{dt} = kP(t)$$

Ecuación malthusiana

El modelo de Lotka-Volterra es el primero de muchos modelos de interacción. Este modelo presenta un comportamiento oscilatorio, que está provocado por una relación depredador-presa con otra especie, cuyo comportamiento es oscilatorio también.

Se define mediante un sistema que incluye las siguientes dos ecuaciones diferenciales ordinarias:

$$\begin{cases} \frac{dP}{dt} = r_1 P - a_1 PD & \text{(ecuación para la población de presas)} \\ \\ \frac{dD}{dt} = a_2 PD - r_2 D & \text{(ecuación para la población de depredadores)} \end{cases}$$

Donde D es el número de depredadores, P es el número de presas y los parámetros son constantes positivas que representan:

- r_1: tasa de crecimiento de las presas.
- a_1: éxito en la caza del depredador, que afecta a la presa.
- r_2: tasa de crecimiento de los depredadores.
- a_2: éxito en la caza, que afecta al depredador.

Gauss–Seidel Method

The **Gauss–Seidel method** updates components immediately:

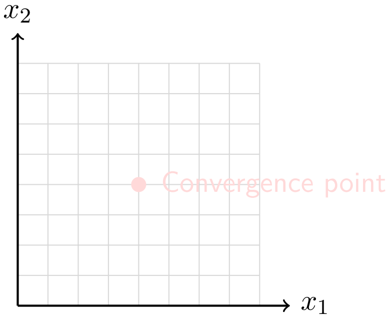
$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right)$$

It often converges faster than Jacobi.

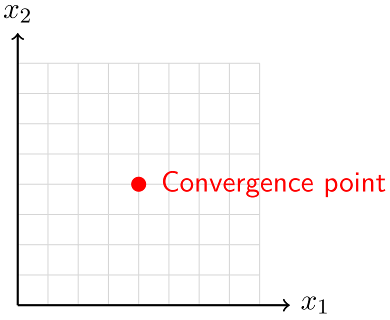
Python Implementation — Gauss–Seidel

```
for k in range(8):
    for i in range(len(A)):
        s1 = sum(A[i,j]*x[j] for j in range(i))
        s2 = sum(A[i,j]*x[j] for j in range(i+1, len(A)))
        x[i] = (b[i] - s1 - s2) / A[i,i]
    print(f"Iteration {k+1}: {x}")
```

Convergence Illustration



Convergence Illustration



Summary

- Jacobi and Gauss–Seidel are basic iterative solvers.
- Convergence requires diagonal dominance or SPD matrices.
- Easy to implement in Python and extend to large systems.
- Serve as a foundation for advanced numerical methods.

Thank You!

Faculty of Systems and Informatics Engineering — UNMSM
Overleaf FISI ▪ flowxy.org

`diego.sotelo@unmsm.edu.pe`