# Simulación de una dinámica poblacional en C++

Mirian Andrea Geronimo Aparicio[1]

[1]Facultad de Ciencias, Universidad Nacional de Ingeniería
[1]mgeronimoa@uni.pe
[1]MirianGeronimo

December 6, 2022

## Resumen

En el presente documento realizaremos la simulación de una dinámica poblacional usando el lenguaje de programación C++ y luego mostraremos la gráfica usando $R$ para determinados valores de captura, tasa de crecimiento poblacional y capacidad de carga.

## 1 Metodología

Dinámica poblacional a simular:

$$N_{t+1} = N_t + rN_t\left(1 - \frac{N_t}{K}\right) - C_t \qquad (1)$$

donde $N_t$ es la abundancia de la población en el tiempo $t$, $C_t$ es la captura en el intervalo $[t, t+1)$, $r$ la tasa de crecimiento poblacional y $K$ la capacidad de carga.
A continuación presentamos la implementación en C++:

```
#include "stdio.h"
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
/*definimos la función captura*/
float C(float t){
return 10;
}
/*definimos la función abundancia de la poblacion en el tiempo t*/
/*que tiene como parámetros el tiempot, población inicial NO*/
/*la tasa r y la capacidad de K*/
float   N(int t, int NO,float r,float K){
float my_vector2[t+1];
int tiempos[t+1];
my_vector2[0]=NO;
tiempos[0]=0;

float N;

for(int i = 0; i<= t-1; i=i+1)
{
N=NO+r*NO*(1-NO/K)-C(i);
NO=N;
my_vector2[i+1]=N;
tiempos[i+1]=i+1;
cout << "La población en tiempo t=" << i+1 << " es N(t)=" << N << endl;


}
/*for(int i = 0; i < t+1; i++){
cout<<my_vector2[i]<<" ";
}*/
ofstream fout("Myfile.txt");
if(fout.is_open())
{
fout << "tiempos" ;
fout << " Poblacion " ;
fout << endl;
for(int i = 0; i<=t; i++)
{
fout << tiempos[i] << " " ;
fout << my_vector2[i] << " " ;
fout << endl;
}
cout << "Success!" << endl;
}
else
{
cout << "File could not be opened." << endl;
}

return my_vector2[t+1];
}

int main()
{
float NO,r,K;
int t;
/*Damos ciertos valores de entrada*/
NO=7;
r=2.4;
K=100;
t=20;
/*imprimimos los resultados*/
printf("La población en el tiempo t=0 es N(t)=%.5f",NO);
printf("\n");
N(t,NO,r,K);
printf("\n");
}
```

Observamos que utilizamos un ciclo for que calculará e imprimirá la abundancia de población para cada $t$ menor igual al valor en el tiempo requerido.
Además se define la función captura $C_t$ y la función $N(int\ t, int\ N0,\ \ float\ r, float\ K)$, esta última retorna el vector llamado $my_vector2$ de longitud $T+1$, donde $T$ es el tiempo en donde se requiere conocer la abundancia de población, el cuál contiene todos los valores que toma $N_t$ para $t \leq T$.
Dentro de esta función utilizamos la clase $ofstream$ para crear un archivo $txt$ en donde podamos almacenar los valores de $N_t$ para cada $t$ y así poder graficar los puntos de la forma $(t, N_t)$ en $R$.
Finalmente en la función principal $int\ main()$ definimos los parámetros de entrada y le damos los valores correspondientes para hallar la dinámica en cierto tiempo $t$.

Ahora presentamos un script en $R$ para graficar los resultados.

```
setwd("/home/mirian/Descargas/claseMrak/elementos finitos")
resul <- read.table(file = "Myfile.txt", header=TRUE)
print(resul)

with(resul,plot(tiempos,Poblacion))
```

## 2 Generación de la simulación

Para poder usar correctamente el código y se tenga una reproducción exitosa de los resultados se recomienda seguir los siguientes pasos:

1. Crear una carpeta en alguna que tenga una ruta de acceso rápida, supongamos que el nombre de la carpeta sea simulacionimarpe.

2. En la carpeta simulacionimarpe guardar el programa que lleva el nombre $problemaIMARPE.cpp$ .

3. Abrir el Visual Studio Code, abrir el programa, correr el código.

4. Automáticamente se creará un archivo txt con el nombre de $Myfile.txt$.

5. Luego, abrir Rstudio, ubicarse en el espacio de trabajo dentro de la carpeta para ello podría usar el comando setwd("*ruta de la carpeta simulacionimarpe*").

6. finalmente correr el script que tiene por nombre $im.r$, luego de ello se generarán las imágenes. También, abriendo el archivo Myfile.txt que se genera con el programa en c++ se puede visualizar la data.

## 3 Resultados

Para verificar la funcionalidad de la implementación proporcionamos ciertos valores a los parámetros:

- $N0 = 7$;
- $r = 2{,}4$;
- $K = 100$;
- $t = 20$;
- $C_t = 10$ (i.e la captura es constante).

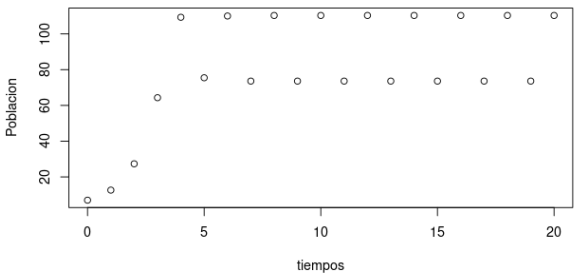


Figura 1: Gráfica de los puntos de la forma $(t, N_t)$ resultantes de la implementación.

```
La población en el tiempo t=0 es N(t)=7.00000
La población en tiempo t=1 es N(t)=12.624
La población en tiempo t=2 es N(t)=27.344
La población en tiempo t=3 es N(t)=64.304
La población en tiempo t=4 es N(t)=109.296
La población en tiempo t=5 es N(t)=75.456
La población en tiempo t=6 es N(t)=110
La población en tiempo t=7 es N(t)=73.6
La población en tiempo t=8 es N(t)=110.304
La población en tiempo t=9 es N(t)=73.6
La población en tiempo t=10 es N(t)=110.304
La población en tiempo t=11 es N(t)=73.6
La población en tiempo t=12 es N(t)=110.304
La población en tiempo t=13 es N(t)=73.6
La población en tiempo t=14 es N(t)=110.304
La población en tiempo t=15 es N(t)=73.6
La población en tiempo t=16 es N(t)=110.304
La población en tiempo t=17 es N(t)=73.6
La población en tiempo t=18 es N(t)=110.304
La población en tiempo t=19 es N(t)=73.6
La población en tiempo t=20 es N(t)=110.304
Success!

[1] + Done                       "/usr/bin/gdb"
icrosoft-MIEngine-Out-wzdiu2us.kun"
(base) mirian@mirian-PC:~/Descargas/claseMrak$
```

Figura 2: Resultados abundancia de la población $N_t$ en el tiempo $t$.

## 4 Discusión de resultados

- Usamos la función de carga $C_t$ constante, fácilmente se podría editar el algoritmo y cambiar hacia una función que dependa del $t$.
- En la figura 1 observamos cierto movimiento en zigzag a partir del tiempo $t = 5$.

## Conclusiones

- Se obtuvo la simulación para la correspondiente dinámica poblacional.
- Resultó más sencillo y aprovechable exportar los datos a $R$ para realizar las gráficas.