Práctica 1 - Sistemas de Tiempo Discreto

Modelado y Simulación de Sistemas Dinámicos

Mateo Bacci, Sebastian Giulianelli, Agustina Torrano

3 de junio de 2025

Problema 1

1. Modelo de *Ecuaciones en Diferencias* que expresa la dinámica de $P(t)$:

$$P(t+1) = b \,.\, P(t) - d \,.\, P(t)$$

2. Función escrita en Octave que permite simular el modelo a partir de una condición inicial dada:

```
function [t,P] = poblacion(P0, ti, tf)
    b = 0.1;
    d = 0.02;
    t = [ti:tf];
    p=zeros(1, length(t));
    P(1) = P0;
    for k=[1:length(t)-1]
        P(k+1) = P(k) + b*P(k) - d*P(k);
    end
end
```

Al simular el modelo con la implementación anterior se obtiene el gráfico 1.

3. Para que la tasa de mortalidad sea proporcional al número de individuos basta con modificar el código anterior de la siguiente manera:

```
function [t,P] = poblacion2(P0, ti, tf)
  b = 0.1;
  a = 0.02;
  t = [ti:tf];
  p=zeros(1, length(t));
  P(1) = P0;
  for k=[1:length(t)-1]
    P(k+1) = P(k) + b*P(k) - a*P(k)*P(k);
  end
end
```

4. El gráfico que se obtiene al ejecutar esta nueva versión del modelo se muestra en la figura 2.

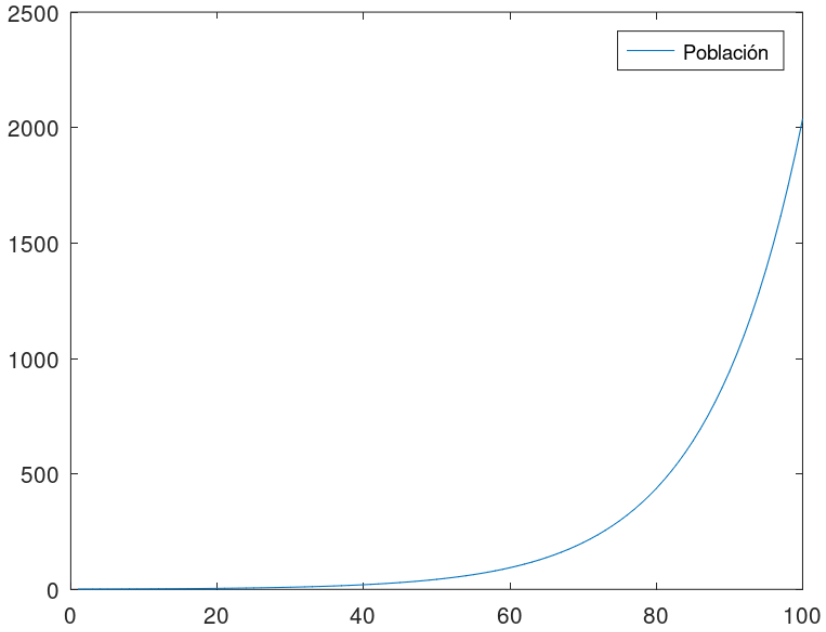

Figura 1: Simulación del crecimiento de una población con tasas de natalidad ($b = 0{,}1$) y mortalidad ($d = 0{,}02$) constantes, y población inicial igual a 1.

5. A partir de la última modificación, el cálculo de $P(t+1)$ es $P(t+1) = P(t) + b\,.\,P(t) - a\,.\,P(t)^2$. Como los puntos de equilibrio son aquellos donde $P(t+1) = P(t)$, para este caso se debe cumplir que $b\,.\,P(t) - a\,.\,P(t)^2 = 0$. Como a y b son constantes positivas:

$$\begin{aligned} b\,.\,P(t) - a\,.\,P(t)^2 &= 0 \\ b\,.\,P(t) &= a\,.\,P(t)^2 \\ b &= a\,.\,P(t) \quad (P(t) \neq 0) \\ P(t) &= \frac{b}{a} \end{aligned} \tag{1}$$

Luego, con $a = 0{,}01$ y $b = 0{,}1$, el punto de equilibro del sistema es $\frac{0{,}1}{0{,}01} = 10$, como se observa en la figura 2.

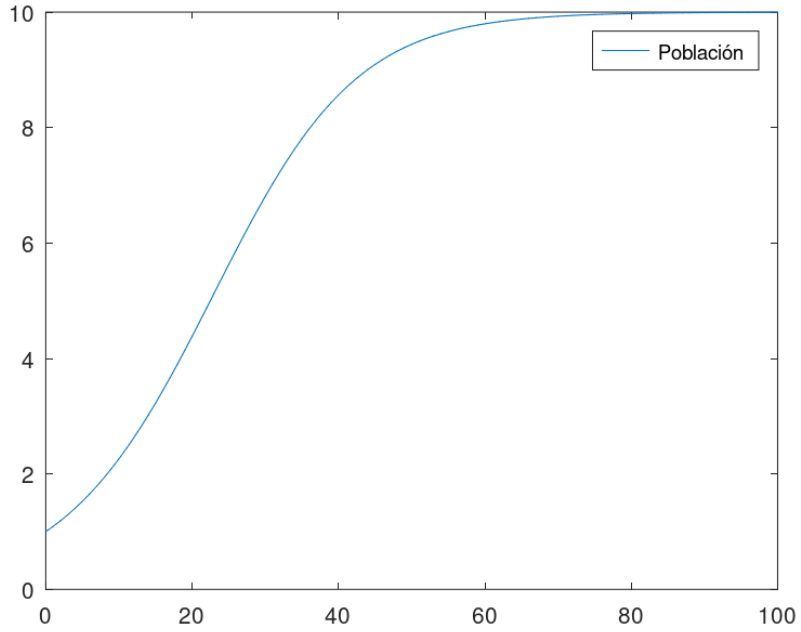

Figura 2: Simulación del crecimiento de una población tasa de natalidad constante ($b = 0{,}1$), tasa de mortalidad proporcional al número de individuos ($a = 0{,}01$) y población inicial igual a 1.

Problema 2

1. Función en Octave que permite simular un modelo epidemiológico SIR de tiempo discreto:

```
function [t,x] = SIR(N, al, gam, S0, I0, ti, tf)
 t = [ti:tf];
 S = zeros(1, length(t));
 I = zeros(1, length(t));
 R = zeros(1, length(t));
 S(1) = S0;
 I(1) = I0;
 R(1) = 0;
 for k=[1:length(t)-1]
  S(k+1) = S(k)- al * S(k) * I(k) / N;
  I(k+1) = I(k) + al * S(k) * I(k) / N - gam * I(k);
  R(k+1) = R(k) + gam * I(k);
 endfor
 x=[S;I;R];
endfunction
```

   Al simular el modelo con la implementación anterior se obtiene el gráfico 3.

2. Al repetir el punto anterior con las nuevas funciones, el resultado obtenido es el mismo, sin embargo, separar el problema en dos funciones permite mayor flexibilidad al momento de aplicar modificaciones en el futuro.

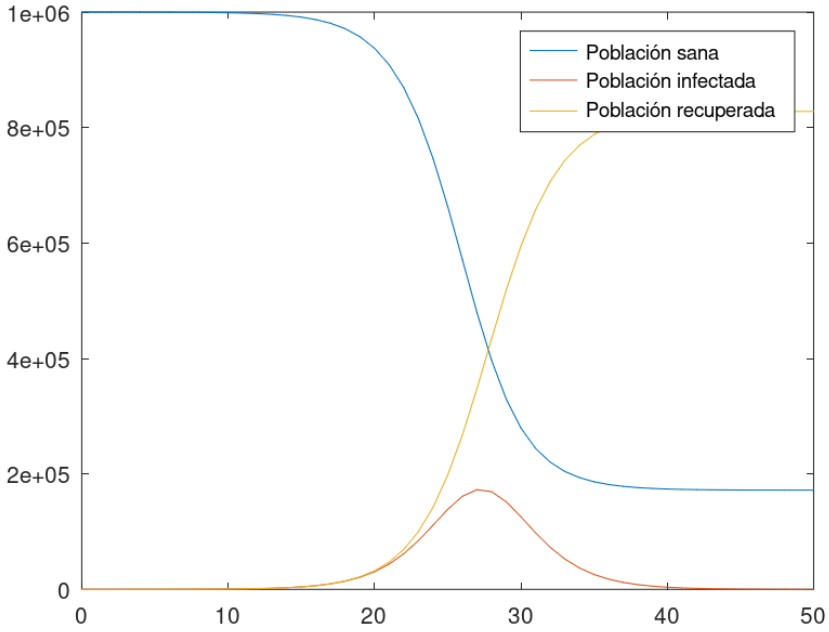

Figura 3: Simulación de un modelo epistemiológico SIR de tiempo discreto con parámetros $N = 10^6$, $\alpha = 1$, $\gamma = 0{,}5$, y condición inicial $S(0) = 10^6$ y $I(0) = 10$.

Problema 3

1. Gracias al punto 2 del ejercicio anterior, para este problema sólo basta con implementar el algoritmo del modelo SEIR y usar el algoritmo de simulación ya implementado. Para ello, definimos la función `discreteSEIR` como una modificación de la función `discreteSIR` anterior:

```
function x=discreteSEIR(pre_x,t)
  al=1;
  gam=0.5;
  mu = 0.5
  N=1e6;
  pre_S=pre_x(1);
  pre_I=pre_x(2);
  pre_R=pre_x(3);
  pre_E=pre_x(4);
  S = pre_S - al * pre_S * pre_I / N;
  I = pre_I + mu*pre_E - gam * pre_I;
  R = pre_R + gam * pre_I;
  E = pre_E + al * pre_S * pre_I / N - mu*pre_E;
  x=[S;I;R];
endfunction
```

Al simular el modelo con la implementación anterior se obtiene el gráfico 4.

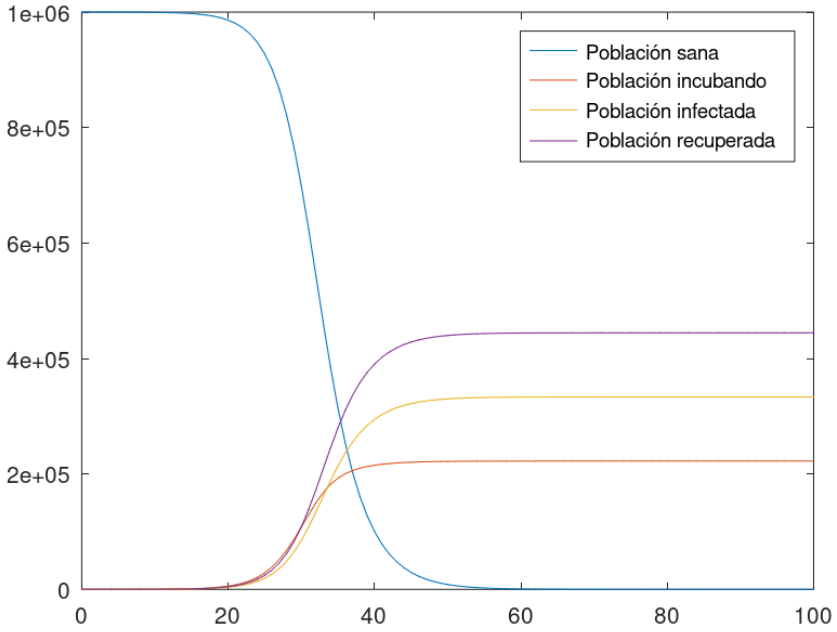

Figura 4: Simulación de un modelo epistemiológico SEIR de tiempo discreto con parámetros $N = 10^6$, $\alpha = 1$, $\gamma = 0{,}5$, $\mu = 0{,}5$ y condición inicial $S(0) = 10^6$ y $I(0) = 10$.

Problema 4

1.

$$
\begin{aligned}
N_0^E(t) &= \frac{R_0}{TR - TI} \cdot \frac{I(t) \cdot S(t)}{N} \\
N_1^E(t+1) &= N_0^E(t) \\
N_2^E(t+1) &= N_1^E(t) \\
&\vdots \\
N_{TR}^E(t+1) &= N_{TR-1}^E(t) \\
S(t+1) &= S(t) - N_0^E(t) \\
E(t+1) &= E(t) + N_0^E(t) - N_{TI}^E(t) \\
I(t+1) &= I(t) + N_{TI}^E(t) - N_{TR}^E(t) \\
R(t+1) &= R(t) + N_{TR}^E(t)
\end{aligned}
$$

2. La implementación en Octave del modelo SEIR a tiempo discreto con retardos explícitos es la siguiente:

```
function x = discreteSEIR2(pre_x, t)
  TR=length(pre_x)-4;
  TI=3, N=1e6;
  R0=1.5;
  pre_S=pre_x(1);
  pre_E=pre_x(2);
  pre_I=pre_x(3);
  pre_R=pre_x(4);
```

```
  pre_NE=pre_x(5:length(pre_x))
  NEaux=R0/(TR-TI) * pre_S * pre_I / N;
  S=pre_S - NEaux;
  E=pre_E + NEaux - pre_NE(TI);
  I=pre_I + pre_NE(TI) - pre_NE(TR);
  R=pre_R + pre_NE(TR);
  NE=[NEaux; pre_NE(1:TR-1)]
  x=[S;E;I;R;NE]
end
```

Al simularlo con la implementación del algoritmo de simulación del ejercicio 2.2, obtenemos la gráfica de la figura 12.

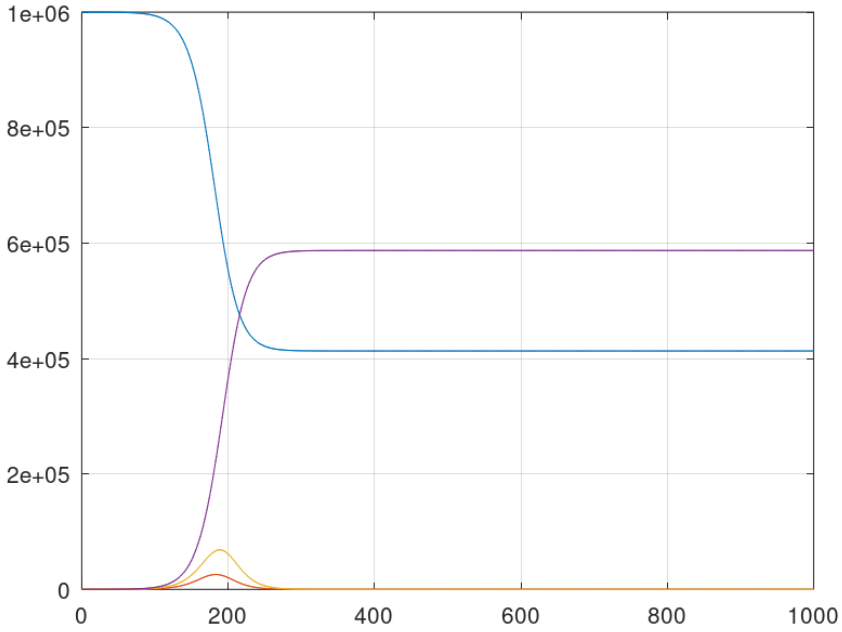

Figura 5: Simulación de un modelo epistemiológico SEIR de tiempo discreto con retardos explícitos. Parámetros usados: $T_I = 3$, $T_R = 12$ y $R_0 = 15$.

Problema 5

1. Modelo SIR en OpenModélica

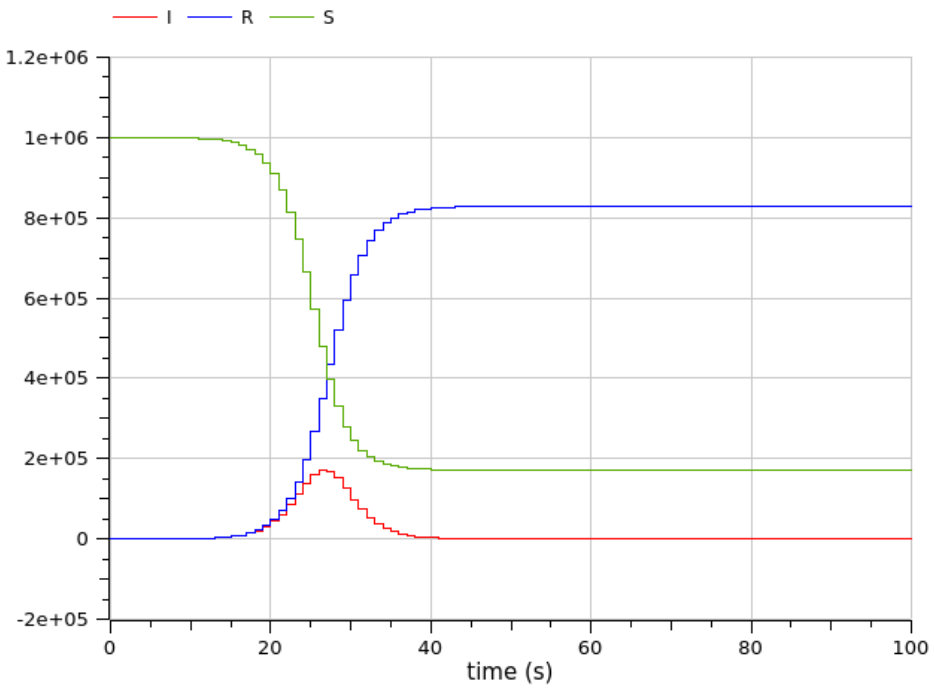

Figura 6: Simulación del modelo epistemiológico SIR en openModélica.

2. La implementación en OpenModélica del modelo SEIR de tiempo discreto con retardos explícitos es la siguiente:

```
model discreteSEIR2
  parameter Real N = 1e6;
  discrete Real S(start = N), I(start = 10), R(start = 0), E(start = 0),
      NEaux;
  discrete Real NE[12](start=cat(1, {10.0}, fill(0.0, 11)));

  parameter Real alpha = 1, gamma = 0.5, mu = 0.5;
  parameter Integer Ti = 3, Tr = 12;
  parameter Real R0 = 1.5;

algorithm
  when sample(0, 1) then
    NEaux := (R0 / (Tr - Ti)) * (pre(S) * pre(I) / N);

    S := pre(S) - NEaux;
    E := pre(E) + NEaux - pre(NE[Ti]);
    I := pre(I) + pre(NE[Ti]) - pre(NE[Tr]);
    R := pre(R) + pre(NE[Tr]);

    NE := cat(1, {NEaux}, pre(NE[1:11]));
  end when;
end discreteSEIR2;
```

Al simularlo obtenemos la gráfica de la figura 12.

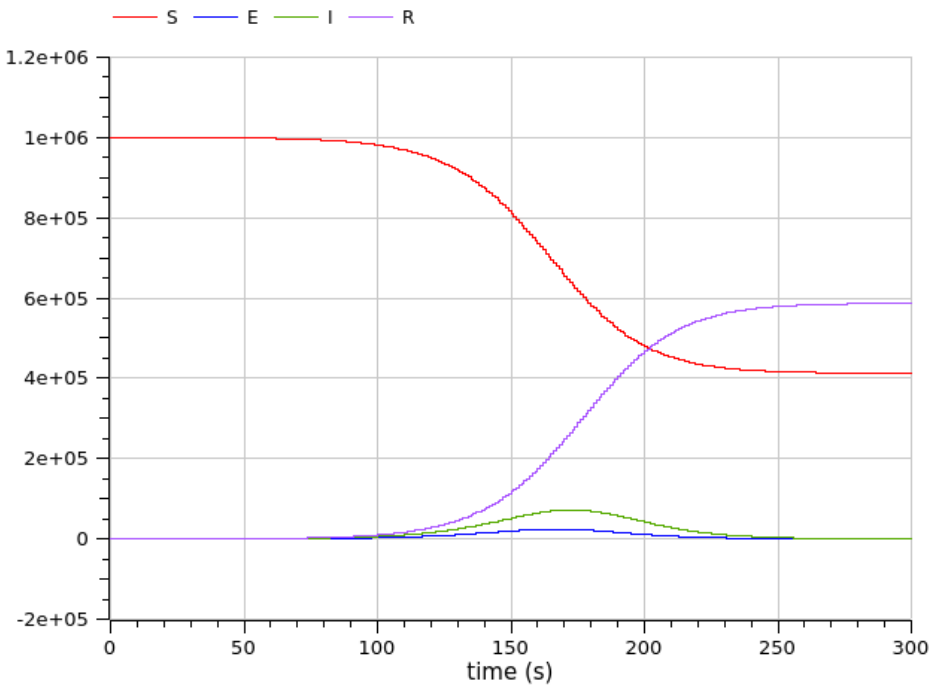

Figura 7: Simulación de un modelo epistemiológico SEIR de tiempo discreto con retardos explícitos. Parámetros usados: $T_I = 3$, $T_R = 12$ y $R_0 = 1{,}5$.

Problema 6

1. Modelo multisir

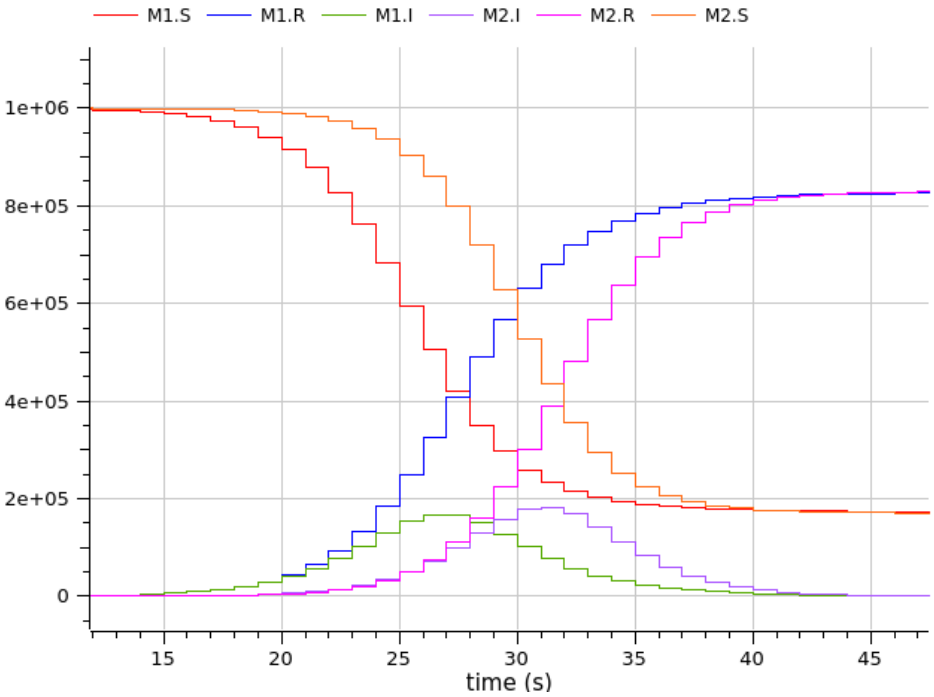

Figura 8: Simulación del modelo de dos poblaciones en openModélica.

2. La implementación en OpenModélica del modelo de tres poblaciones es la siguiente:

```
model multiSIR2
  discreteSIRimp M1, M2(I.start = 0), M3(I.start=0);
algorithm
  when sample(0, 1) then
    M1.imp := pre(M2.exp) / 2;
    M3.imp := pre(M2.exp) / 2;
    M2.imp := pre(M1.exp) + pre(M3.exp);
  end when;
end multiSIR2;
```

Al simularlo obtenemos la gráfica de la figura 12.

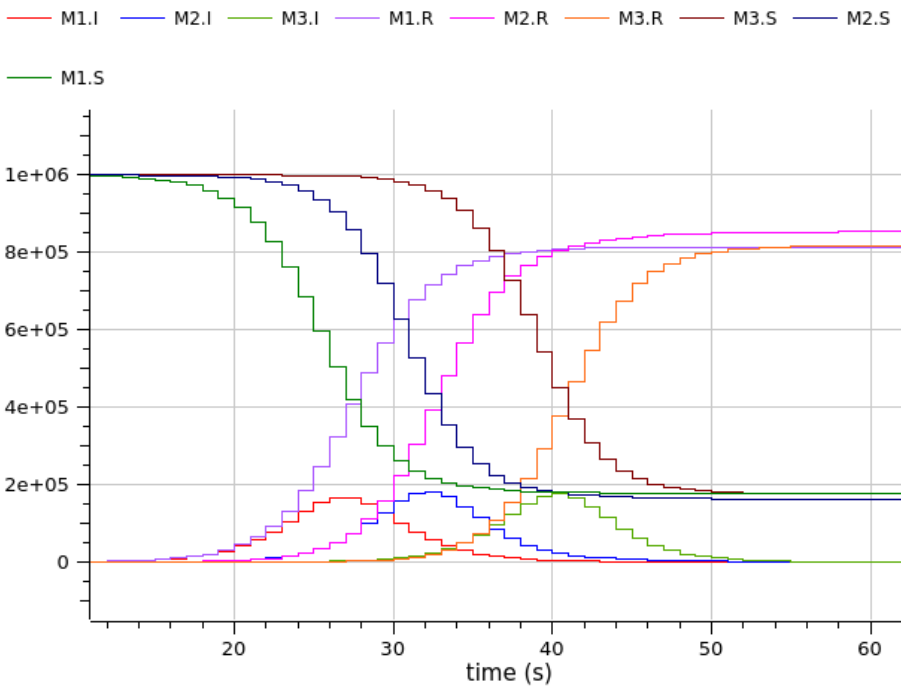

Figura 9: Simulación del modelo de tres poblaciones.

3. Implementar estos modelos como funciones de Octave resultaría difícil ya que se requiere sincronización y comunicación entre los modelos. Como Octave no esta diseñado con ese objetivo, habría que implementarlo todo e incluso se llegaría a una solución poco escalable y sencilla.

Problema 10

1. La implementación en OpenModélica del Modelo en Tiempo Discreto y Control de un Robot Móvil es la siguiente:

```
model Problema10
  discrete Real x, y, th, v, psi, a, w;
  parameter Real L=1, h=0.1;
algorithm
```

```
  when sample(0,h) then
    x:=pre(x) + h*cos(pre(th))*cos(pre(psi))*pre(v);
    y:= pre(y) + h*sin(pre(th))*cos(pre(psi))*pre(v);
    th := pre(th)  + h*sin(pre(psi))*pre(v)/L;
    v:= pre(v) + h*pre(a);
    psi:= pre(psi) + h*pre(w);
    if time<1 then
      a:=1;
      w:=0.1;
    else if time >= 2 then
      a:= 0;
      w:= 0;
    else
      a:= 1;
      w:=-0.1;
    end if;
    end if;
end when;
end Problema10;
```

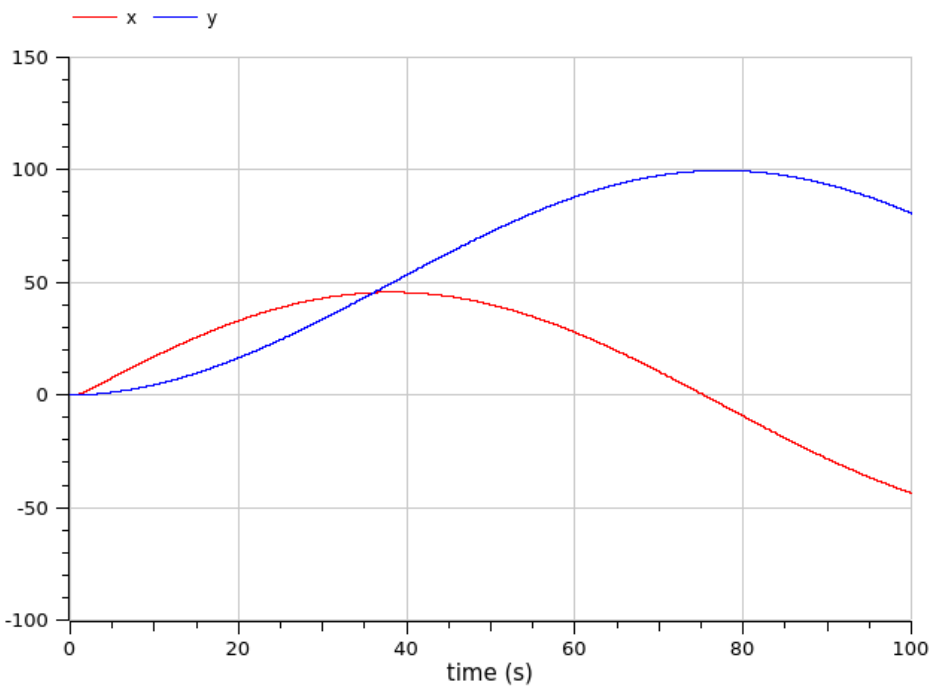

Figura 10: Simulación del modelo del robot móvil.

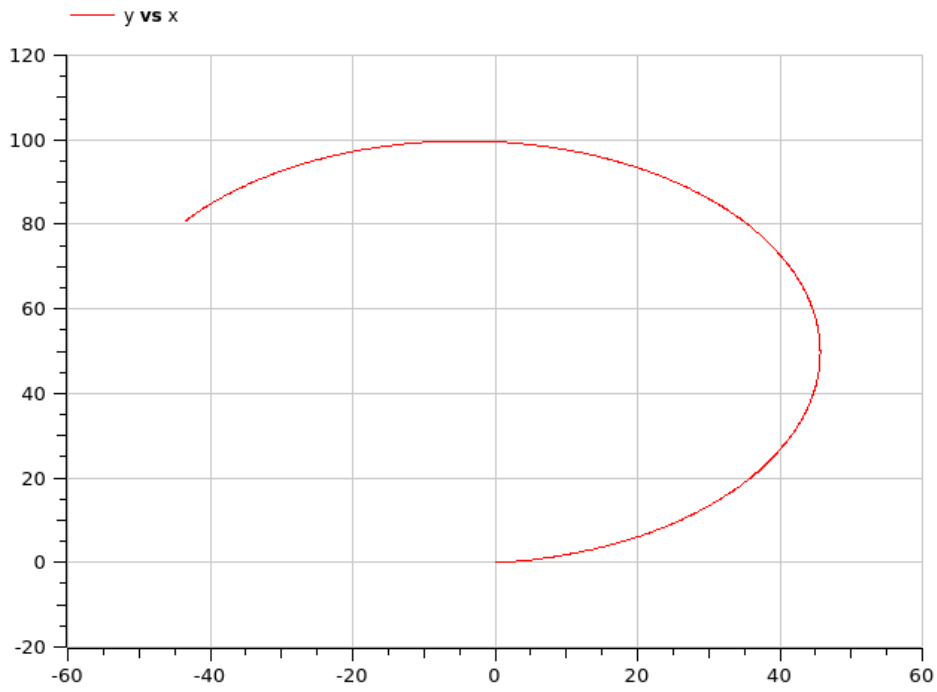

Figura 11: Trayectoria marcada por el robót durante la simulación.

Se puede observar que la trayectoria que marca el robot aparenta ser una circunferencia y esto es debido a que las variables que miden el giro y la velocidad del robot están fuertemente afectadas por los valores a y w, que después de algunas unidades de tiempo se hacen 0 y en consecuencia, la rotación y la velocidad del robot se vuelven constantes a lo largo del tiempo lo que genera una curva suave.

2. La implementación de la segunda ley de control es la siguiente:

```
model Problema10b
  discrete Real x, y, th, v, psi, a, wpsi, psiref;
  parameter Real L=1, h=0.1;
  parameter Real Kv = 1, Kpsi = 2, Ky = 0.5, L0 = 4, vref = 2, yref = 1;
algorithm
  when sample(0,h) then
    x:=pre(x) + h*cos(pre(th))*cos(pre(psi))*pre(v);
    y:= pre(y) + h*sin(pre(th))*cos(pre(psi))*pre(v);
    th := pre(th)  + h*sin(pre(psi))*pre(v)/L;
    v:= pre(v) + h*pre(a);
    psi:= pre(psi) + h*wpsi;
    a:= Kv * (vref - pre(v));
    wpsi := Kpsi * (psiref - pre(psi));
    psiref := Ky * (atan(yref - pre(y)) / L0 - pre(th));
  end when;
end Problema10b;
```

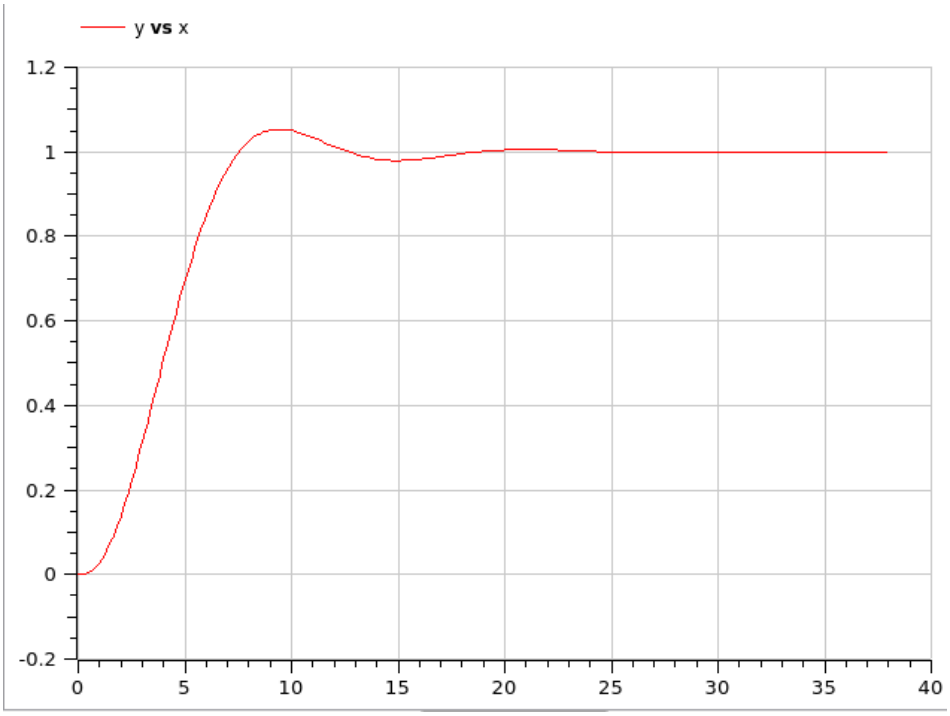

Figura 12: Trayectoria marcada por el robot durante la simulación.

Vemos que en este caso la trayectoria del robot ya no marca una curva tan marcada como en el primer caso. Esto ocurre debido a que el robot corrige su dirección constantemente para alinearse con la linea de referencia.