Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Licenciatura en Ciencias de la Computación

Entrega 1

Modelado y Simulación de Sistemas Dinámicos

Arroyo Joaquín
Bolzan Francisco

Contents

Notar que los problemas que piden ser realizados en *Octave* fueron realizados en *Scilab*

1 Problema 1

Crecimiento de una Población. En ausencia de limitaciones ambientales, muchas especies se reproducen de manera tal que el nacimiento de individuos es proporcional al tamaño de la población. Llamaremos $P(t)$ a dicho tamaño, b a la tasa de natalidad (número de nacimientos por unidad de tiempo) y d a la tasa de mortalidad (número de fallecimientos por unidad de tiempo). Tanto b como d estarán expresados como fracción del tamaño actual de la población (es decir, $b = 0.1$ implica que el número de nacimientos por unidad de tiempo es un 10% del tamaño de la población).

Se pide lo siguiente:

1. Obtener un modelo de Ecuaciones en Diferencias que exprese la dinámica de $P(t)$

$$P(t+1) \;=\; P(t) + b \cdot P(t) - d \cdot P(t)$$

2. Escribir una función que permita simular el modelo a partir de una condición inicial arbitraria. Observar el resultado que se obtiene con $b = 0.1$ y $d = 0.02$ para $P(0) = 1$.

```
function x = sim(x0, tf)
    b = 0.1;
    d = 0.02;
    x = zeros(1, tf);
    x(1) = x0;
    for k = 1:tf-1
        x(k+1) = x(k)*(1 + b - d)
    end
end
```

3. Modificar el modelo para contemplar limitaciones ambientales, suponiendo ahora que la tasa de mortalidad es proporcional al número de individuos, es decir, $d(t) = a \cdot P(t)$ para cierta constante $a > 0$.

$$P(t+1) \;=\; P(t) + b \cdot P(t) - d(t) \cdot P(t) = P(t) + b \cdot P(t) - a \cdot P(t)^2$$

```
function x = dtsim(f, x0, tf)
  x = zeros(1, tf);
  x(:,1) = x0;
  for k = 1:tf-1
    x(:,k+1) = f(x(:,k), k);
  end
end

function xk1 = population(xk, k)
    b = 0.1;
    a = 0.01;
    d = a * xk;
    xk1 = xk + b * xk - d * xk;
end
```

4. Simular este nuevo modelo con $a = 0.01$, $b = 0.1$, $P(0) = 1$.

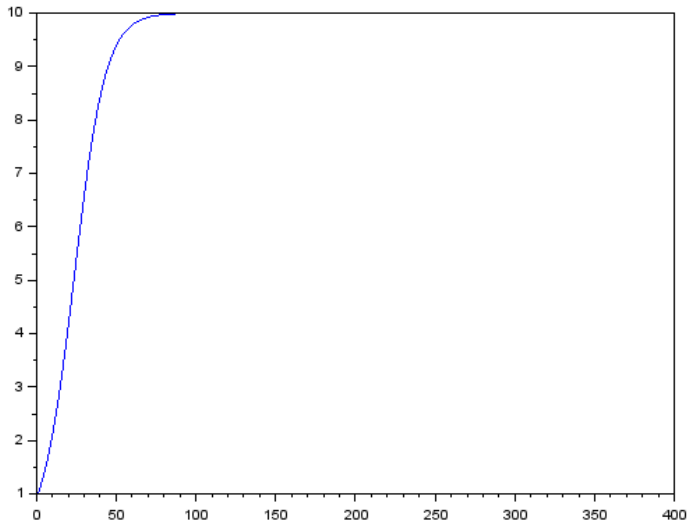

Figure 1: Simulación del modelo con $a = 0.01$, $b = 0.1$, $P(0) = 1$

5. Para este último caso, calcular analíticamente los puntos de equilibrio.
   Queremos ver donde $P(t+1) = P(t)$, viendo la ecuación, esto sucede si

$$b \cdot P(t) - a \cdot P(t)^2 = 0.$$

   Luego, esto vale para $x_1 = 0$ y $x_2 = 10$.

2 Problema 2

Modelo Epidemiológico SIR discreto. El siguiente modeloes una versión de tiempo discreto de un modelo clásico de epidemiología conocido como SIR (*Susceptible, Infected, Recovered*).

$$S(t+1) = S(t) - \frac{\alpha \cdot S(t) \cdot I(t)}{N}$$
$$I(t+1) = I(t) + \frac{\alpha \cdot S(t) \cdot I(t)}{N} - \gamma \cdot I(t)$$
$$R(t+1) = R(t) + \gamma \cdot I(t)$$

Las variables $S(t)$, $I(t)$ y $R(t)$ representan el número de individuos susceptibles, infectados y recuperados, respectivamente. Los parámetros, en tanto, son N (población total), α (tasa de contacto) y γ (tasa de recuperación). α expresa el número de individuos en promedio por día con los que cada individuo infectado está en suficiente contacto como para infectarlo.

Se pide entonces

1. Programar en una función que permita simular este modelo desde condiciones iniciales arbitrarias. Simular el modelo con parámetros $N = 10^6, \alpha = 1, \gamma = 0.5$ para la condición inicial $S(0) = N$, $I(0) = 10$.

```
function [t, x] = dtsim(f, x0, ti, tf)
    t = [ti:tf];
    x(:,1) = x0;
    for k = 1:length(t)-1
       x(:,k+1) = f(x(:,k), t(k));
    end
end

function x = discreteSIR(pre_x, t)
    al=1;
    gam=0.5;
    N=1e6;
    pre_S=pre_x(1);
    pre_I=pre_x(2);
    pre_R=pre_x(3);
    S = pre_S - (al * pre_S * pre_I) / N;
    I = pre_I + ((al * pre_S * pre_I) / N) - gam * pre_I;
    R = pre_R + gam * pre_I;
    x=[S;I;R];
end
```

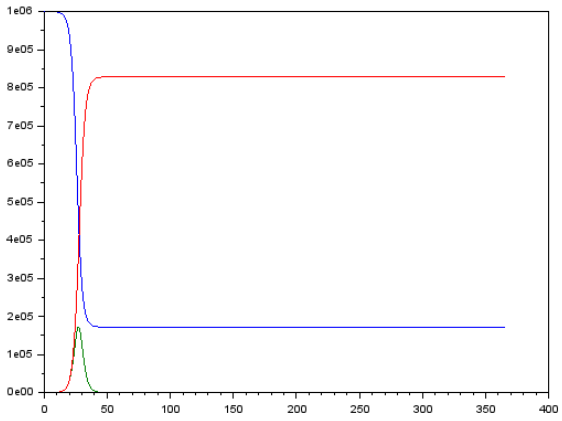

Figure 2: Simulación del modelo con parámetros $N = 10^6, \alpha = 1, \gamma = 0.5$ para la condición inicial $S(0) = N$, $I(0) = 10$, $R(0) = 0$, $b = 0.1$,

3 Problema 3

Modelo SEIR discreto. Una modificación del modelo anterior surge al considerar que en ciertas enfermedades las personas no comienzan a contagiar apenas se exponen al virus, sino que hay un tiempo de incubación. Esto agrega una variable más al modelo, $E(t)$, que representa el número de individuos expuestos que aún no contagian:

$$S(t+1) = S(t) - \frac{\alpha \cdot S(t) \cdot I(t)}{N}$$
$$E(t+1) = E(t) + \frac{\alpha \cdot S(t) \cdot I(t)}{N} - \mu \cdot E(t)$$
$$I(t+1) = I(t) + \mu \cdot E(t) - \gamma \cdot I(t)$$
$$R(t+1) = R(t) + \gamma \cdot I(t)$$

Se propone entonces modificar el ejemplo anterior para considerar este nuevo modelo y simularlo con el parámetro $\mu = 0.5$.

Notar que se utiliza la misma función *dtsim* que en el problema 2.

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

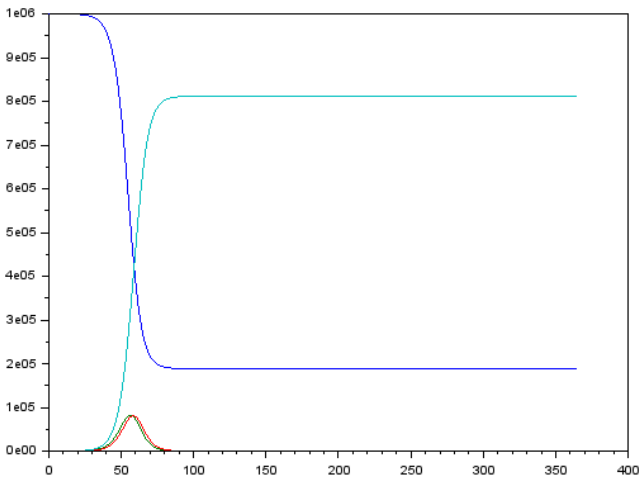

Figure 3: Simulación del modelo con parámetros $\alpha = 1$, $\gamma = 0.5$, $\mu = 0.5$, $N = 10^6$, para la condición inicial $S(0) = N$, $E(0) = 0$, $I(0) = 10$, $R(0) = 0$, $b = 0.1$

4 Problema 4

Modelo SEIR discreto con Retardos Explícitos. Una mejora al modelo anterior surge al considerar explícitamente el tiempo que transcurre desde que cada individuo se expone hasta que se torna infeccioso y hasta que se recupera, lo que lleva al siguiente sistema de ecuaciones:

$$
\begin{aligned}
N^E(t) &= \tfrac{R_0}{T_R - T_I} \cdot \tfrac{I(t)\cdot S(t)}{N} \\
S(t+1) &= S(t) - N^E(t) \\
E(t+1) &= E(t) + N^E(t) - N^E(t - T_I) \\
I(t+1) &= I(t) + N^E(t - T_I) - N^E(t - T_R) \\
R(t+1) &= R(t) + N^E(t - T_R)
\end{aligned}
$$

donde T_I y T_R son los tiempos (enteros) hasta la infección y recuperación, respectivamente, y R_0 es el número reproductivo básico (cuantos individuos son expuestos al virus por cada infectado). Notar que este modelo no está escrito en la forma típica: $x(t+1) = f(x(t), t)$, ya que aparecen dependencias con valores anteriores de la variable $N^E(t)$.

Se pide entonces:

1. Re-escribir las ecuaciones como $x(t+1) = f(x(t), t)$. Para esto, pueden tomarse como variables de estado $N_i^E(t) = N^E(t - i)$ para $i = 1, 2, \cdots, T_R$.

$$
\begin{aligned}
N^E(t) &= \tfrac{R_0}{T_R - T_I} \cdot \tfrac{I(t)\cdot S(t)}{N} \\
N_{T_I}^E(t) &= N^E(t - T_I) \\
N_{T_R}^E(t) &= N^E(t - T_R) \\
S(t+1) &= S(t) - N^E(t) \\
E(t+1) &= E(t) + N^E(t) - N_{T_I}^E(t) \\
I(t+1) &= I(t) + N_{T_I}^E(t) - N_{T_R}^E(t) \\
R(t+1) &= R(t) + N_{T_R}^E(t)
\end{aligned}
$$

2. Implementar el modelo correspondiente y simularlo con parámetros $T_I = 3$, $T_R = 12$ y $R_0 = 1.5$.

```
function [t, x] = dtsim(f, x0, ti, tf)
    t = [ti:tf];
    x(:,1) = x0;
    for k = 1:length(t)-1
       x(:,k+1) = f(x, k);
    end
end

function x = discreteSEIRD(x, k)
    TI=3;
    TR=12;
    R0=1.5;
    N=1e6;
    pre_x = x(:,k);
    pre_S=pre_x(1);
    pre_E=pre_x(2);
    pre_I=pre_x(3);
    pre_R=pre_x(4);
    NET = calculateNET(x, k, TI, TR, R0, N);
    NETR = calculateNET(x, k - TR, TI, TR, R0, N);
    NETI = calculateNET(x, k - TI, TI, TR, R0, N);
```

```
    S = pre_S - NET;
    E = pre_E + NET - NETI;
    I = pre_I + NETI - NETR;
    R = pre_R + NETR;
    x=[S;E;I;R];
end

function x = calculateNET(x, k, TI, TR, R0, N)
    if k < 1
    then x = 0;
    else
        pre_S = x(:,k)(1);
        pre_I = x(:,k)(3);
        x = (R0 / (TR - TI)) * ((pre_I * pre_S) / N);
    end
endfunction
```

Notar que se necesitó hacer una nueva función *dtsim* debido a que en este caso necesitamos acceso no solo al estado anterior para calcular al siguiente, si no que en algunos casos dependemos de las constantes T_I y T_R.

Esta nueva función recibe todos los estados anteriores al momento $t + 1$.

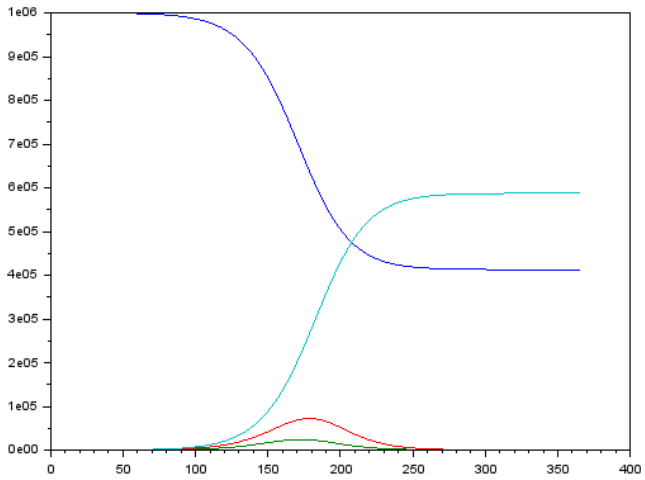

Figure 4: Simulación del modelo con parámetros $T_I = 3$, $T_R = 12$, $R_0 = 1.5$, $N = 10^6$, para la condición inicial $S(0) = N$, $E(0) = 0$, $I(0) = 10$, $R(0) = 0$

5 Problema 5

Implementación en Modelica Una alternativa para implementar los modelos anteriores es usar el lenguaje de modelado Modelica. Para el caso del modelo del Problema 2, una posible representación es la siguiente:

```
model discreteSIR
    parameter Real N = 1e6;
    discrete Real S(start = N), I(start = 10), R;
    parameter Real alpha = 1, gamma = 0.5;
algorithm
    when sample(0, 1) then
        S := pre(S) - alpha * pre(S) * pre(I) / N;
        I := pre(I) + alpha * pre(S) * pre(I) / N - gamma * pre(I);
        R := pre(R) + gamma * pre(I);
    end when;
end discreteSIR;
```

1. Implementar este modelo en la herramienta OpenModelica y simularlo.

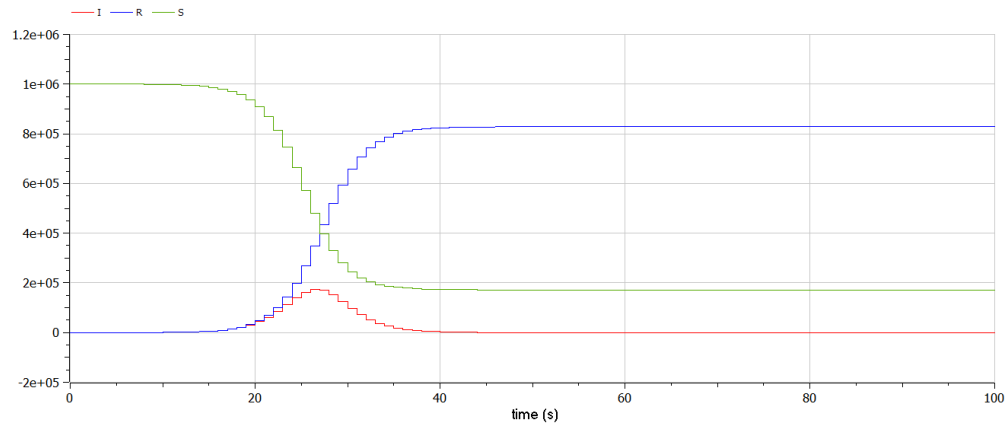

Figure 5: Simulación del modelo con parámetros $N = 10^6, \alpha = 1, \gamma = 0.5$ para la condición inicial $S(0) = N$, $I(0) = 10$, $R(0) = 0$, $b = 0.1$,

2. Implementar los modelos de los Problemas 3-4 en Modelica

```
model discreteSEIR
    parameter Real alpha = 1, gamma = 0.5, mu = 0.5, N = 1e6, b = 0.1;
    discrete Real S(start = N), I(start = 10), R(start = 0), E(start = 0);
algorithm
    when sample(0, 1) then
        S := pre(S) - alpha * pre(S) * pre(I) / N;
        E := pre(E) + alpha * pre(S) * pre(I) / N - mu * pre(E);
        I := pre(I) + mu * pre(E) - gamma * pre(I);
        R := pre(R) + gamma * pre(I);
    end when;
end discreteSEIR;
```

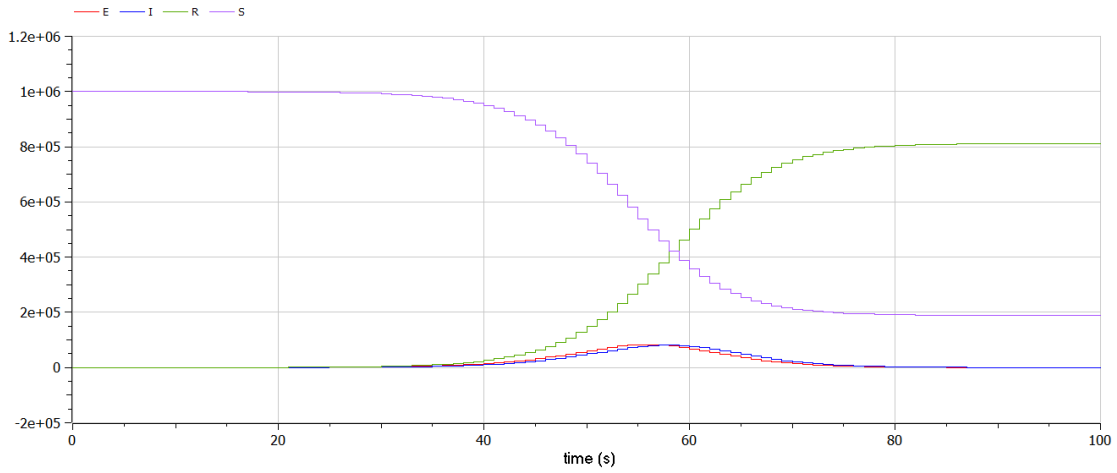

Figure 6: Simulación del modelo del Problema 3 con parámetros $\alpha = 1$, $\gamma = 0.5$, $\mu = 0.5$, $N = 10^6$, para la condición inicial $S(0) = N$, $E(0) = 0$, $I(0) = 10$, $R(0) = 0$, $b = 0.1$

```
model discreteSEIRD
    parameter Integer TI=3, TR=12;
    parameter Real R0=1.5, N=1e6, Cte=R0/(TR-TI);
    discrete Real S(start = N), I(start = 10), R(start = 0), E(start = 0);
    discrete Real NE[500], NETI, NETR;
    discrete Integer C(start = 1);
algorithm
    when sample(0, 1) then
      NE[C] := Cte * ((pre(I) * pre(S)) / N);

      if C-TI >= 1 then
        NETI := NE[C-TI];
      else
        NETI := 0;
      end if;

      if C-TR >= 1 then
        NETR := NE[C-TR];
      else
        NETR := 0;
      end if;

      S := pre(S) - NE[C];
      E := pre(E) + NE[C] - NETI;
      I := pre(I) + NETI - NETR;
      R := pre(R) + NETR;
      C := C+1;
    end when;
end discreteSEIRD;
```

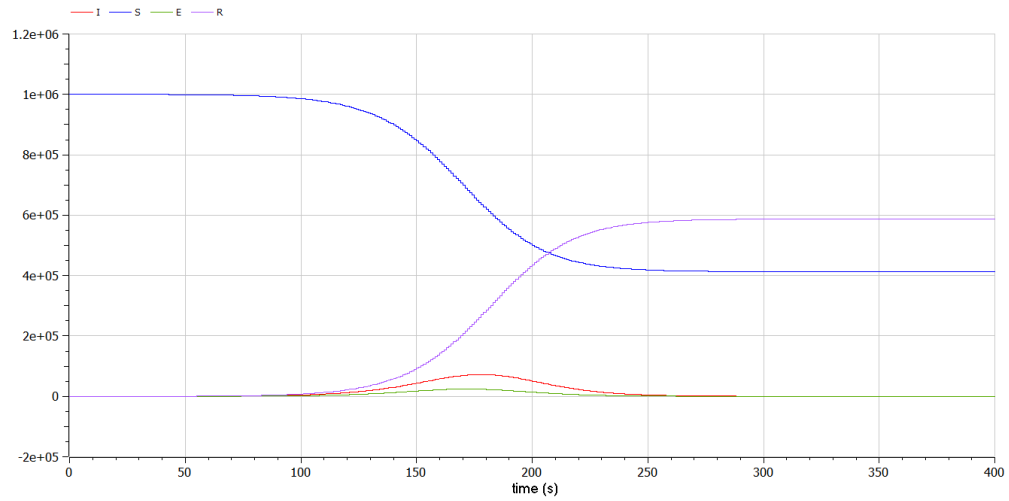

Figure 7: Simulación del modelo del problema 4 con parámetros $T_I = 3$, $T_R = 12$, $R_0 = 1.5$, $N = 10^6$, para la condición inicial $S(0) = N$, $E(0) = 0$, $I(0) = 10$, $R(0) = 0$

6 Problema 6

Modelo de Múltiples Poblaciones El siguiente modelo considera que hay individuos infectados que pueden moverse entre poblaciones:

```
model discreteSIRimp
    parameter Real N = 1e6;
    discrete Real S(start = N), I(start = 10), R, imp, exp;
    parameter Real alpha = 1, gamma = 0.5, m=0.01;
algorithm
    when sample(0, 1) then
        S := pre(S) - alpha * pre(S) * pre(I) / N;
        I := pre(I) + alpha * pre(S) * pre(I) / N - gamma * pre(I) + imp - exp;
        R := pre(R) + gamma * pre(I);
        exp:= m* pre(I);
    end when;
end discreteSIRimp;
```

Aquí, la variable exp es el número de individuos infectados que abandonan la población, mientras que imp son los que llegan. El parámetro m representa la movilidad. Suponiendo ahora que hay dos poblaciones que intercambian individuos según estas reglas, se puede construir el siguiente modelo:

```
model multiSIR
    discreteSIRimp M1, M2(I.start = 0);
algorithm
    when sample(0, 1) then
        M1.imp := M2.exp;
        M2.imp := M1.exp;
    end when;
end multiSIR;
```

1. Implementar en OpenModelica el modelo de dos poblaciones, donde una inicialmente no tiene infectados.

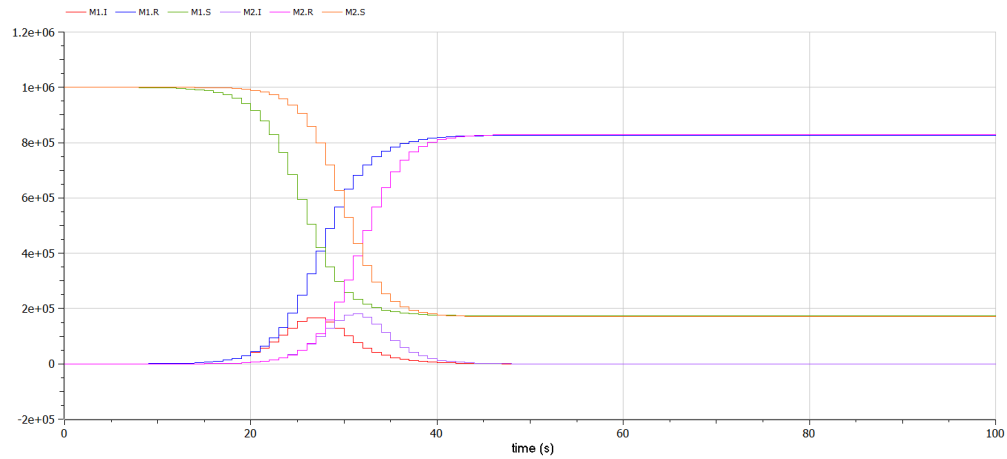

Figure 8: Simulación del modelo con parámetros $\alpha = 1$, $\gamma = 0.5$, $\mu = 0.01$, $N = 10^6$, para la condición inicial $S(0) = N$, $I(0) = 10$, $R(0) = 0$

2. Implementar ahora un modelo de 3 poblaciones de tal manera que la primera intercambie individuos con la segunda, y la segunda lo haga además con la tercera. Suponer que los individuos que salen de la segunda población van la mitad a la 1 y la mitad a la 3.

```
model multiSIR
  discreteSIRimp M1, M2(I.start = 0), M3(I.start = 0);
algorithm
  when sample(0, 1) then
    M1.imp := 0.5 * pre(M2.exp);
    M2.imp := pre(M1.exp) + pre(M3.exp);
    M3.imp := 0.5 * pre(M2.exp);
  end when;
end multiSIR;
```

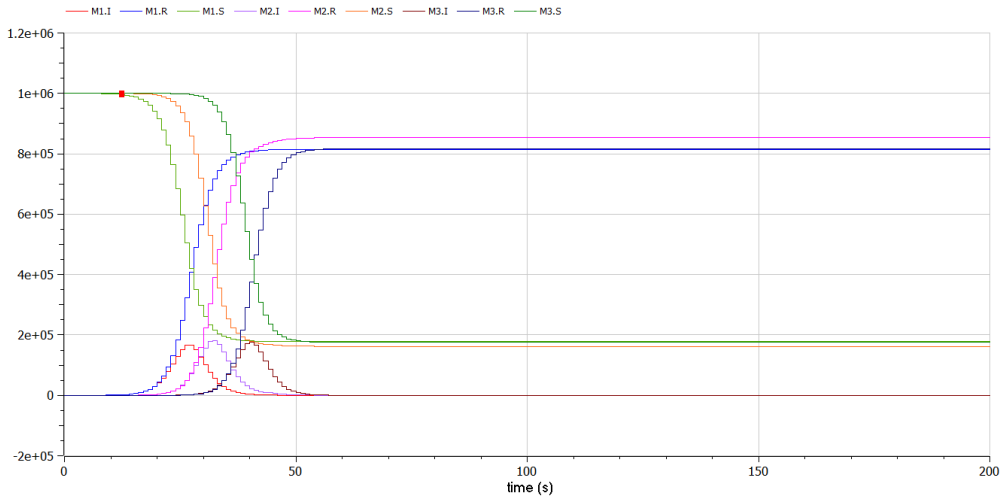

Figure 9: Simulación del modelo anterior con tres poblaciones

3. Analizar los incovenientes que encontraría para implementar estos modelos como funciones.

   Uno de los principales inconvenientes que pensamos es la dificultad de acceder a variables de otros modelos. Al estos estar representados como funciones, las demás funciones no pueden acceder (tan) directamente a sus variables.

7 Problema 10

Modelo en Tiempo Discreto y Control de un Robot Móvil La discretización del modelo de un robot móvil tipo auto que se desplaza en un plano produce el siguiente sistema de ecuaciones en diferencias:

$$x(t+h) = x(t) + h \cdot (cos(\theta) \cdot cos(\psi) \cdot v(t))$$
$$y(t+h) = y(t) + h \cdot (sin(\theta) \cdot cos(\psi) \cdot v(t))$$
$$\theta(t+h) = \theta(t) + h \cdot \frac{(sin(\psi) \cdot v(t))}{L}$$
$$v(t+h) = v(t) + h \cdot a(t)$$
$$\psi(t+h) = \psi(t) + h \cdot \omega_\psi(t)$$

Aquí, $x(t)$ e $y(t)$ representan la posición en el plano del centro del robot, y $\omega(t)$ representa el ángulo que forma el vehículo con el eje horizontal. Por otro lado, $v(t)$ es la velocidad del robot y $\psi(t)$ es el ángulo que forman las ruedas delanteras (dirección) con el robot. El modelo tiene un parámetro L que representa la longitud entre el eje delantero y el trasero y un parámetro h correspondiente al período de la discretización. Además hay dos entradas: la aceleración del vehículo ($a(t)$) y la velocidad de rotación de la dirección $\omega_\psi(t)$.

Se pide entonces

1. Implementar el modelo en PowerDEVS y simularlo suponiendo condiciones iniciales nulas, parámetros $L = 1$, $h = 0.1$, y entradas

$$a(t) = \begin{cases} 1 & si \quad t < 2 \\ 0 & si \quad t \geq 2 \end{cases} \quad \omega_\psi(t) = \begin{cases} 0.1 & si \quad t < 1 \\ -0.1 & si \quad 1 < t < 2 \\ 0 & si \quad t \geq 2 \end{cases}$$

   Interpretar que trayectoria realiza el robot.

```
model robotP1
  parameter Real L=1, h=0.1;
  discrete Real Accel(start = 0), Rotspeed(start = 0);
  discrete Real X(start = 0), Y(start = 0), Vel(start = 0),
    Cangl(start = 0), Wangl(start = 0);
  discrete Real C(start = 0);
algorithm
  when sample(0, h) then
    if C < 2 then
      Accel := 1;
    else
      Accel := 0;
    end if;

    if C < 1 then
      Rotspeed := 0.1;
    elseif C < 2 then
      Rotspeed := -0.1;
    else
      Rotspeed := 0;
    end if;

    X := pre(X) + h * (cos(Cangl)*cos(Wangl)*pre(Vel));
    Y := pre(Y) + h * (sin(Cangl)*cos(Wangl)*pre(Vel));
    Cangl := pre(Cangl) + h * ((sin(Wangl)*pre(Vel))/L);
    Vel := pre(Vel) + h * Accel;
    Wangl := pre(Wangl) + h * Rotspeed;
```

```
    C := C+h;
  end when;
end robotP1;
```

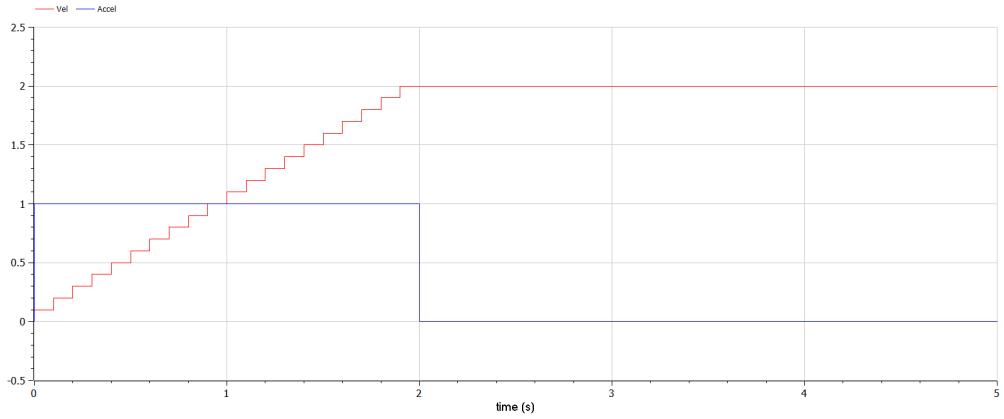

Figure 10: Valores de Velocidad y Aceleración del robot con $L = 1$ y $h = 0.1$

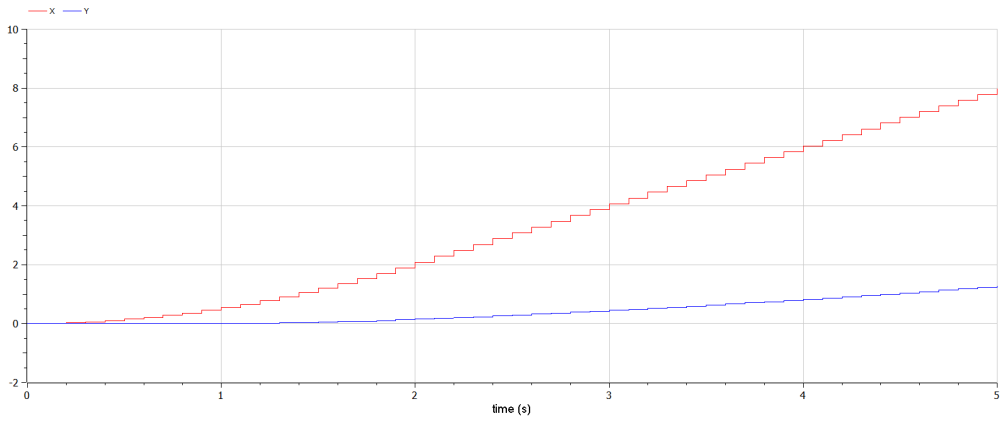

Figure 11: Valores de Posición X e Y del robot con $L = 1$ y $h = 0.1$

Podemos observar entonces que el robot comienza una maniobra de giro hacia su izquierda la cual eventualmente corrige a su posición original para continuar derecho.

2. Suponer ahora que las entradas se calculan según la siguiente *ley de control*

$$a(t) = K_v \cdot (v_{ref} - v(t))$$
$$\omega_\psi(t) = K_\psi \cdot (\psi_{ref} - \psi(t))$$

donde K_v y K_ψ son parámetros de control de velocidad y dirección, v_{ref} es la velocidad deseada y la dirección de referencia se calcula según:

$$\psi_{ref}(t) = K_y \cdot \frac{arctan(y_{ref} - y(t))}{L_0 - \theta(t)}$$

siendo K_y y L_0 parámetros de control de posición, cuya referencia es y_{ref} .

Tomando parámetros $K_v = 1$, $K_\psi = 2$, $K_y = 0.5$, $L_0 = 4$ y las referencias $v_{ref} = 2$ e $y_{ref} = 1$, implementar la ley de control y simular el sistema, interpretando el resultado.

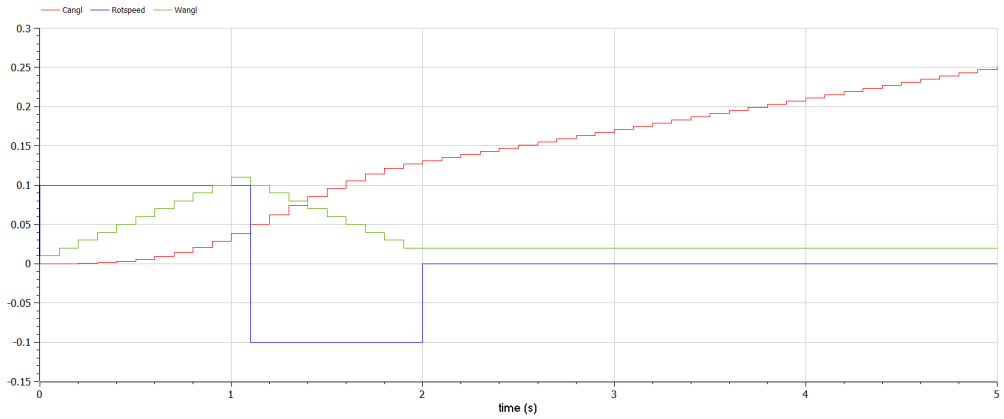

Figure 12: Valores de Angulo del eje del robot y Angulo y Rotación de su dirección con $L = 1$ y $h = 0.1$

```
model robotP2
  parameter Real L=4, h=0.1, Kv=1, Kdir=2, Ky=0.5, vref=2, yref=1;
  discrete Real Accel(start = 0), Rotspeed(start = 0), DirRef(start = 0);
  discrete Real X(start = 0), Y(start = 0), Vel(start = 0),
    Cangl(start = 0), Wangl(start = 0);
  discrete Real C(start = 0);
algorithm
  when sample(0, h) then
    DirRef := Ky * (atan(yref - Y)/(L - Cangl));
    Accel := Kv * (vref - Vel);
    Rotspeed := Kdir * (DirRef - Wangl);

    X := pre(X) + h * (cos(pre(Cangl))*cos(pre(Wangl))*pre(Vel));
    Y := pre(Y) + h * (sin(pre(Cangl))*cos(pre(Wangl))*pre(Vel));
    Cangl := pre(Cangl) + h * ((sin(pre(Wangl))*pre(Vel))/L);
    Vel := pre(Vel) + h * Accel;
    Wangl := pre(Wangl) + h * Rotspeed;

    C := C+h;
  end when;
end robotP2;
```

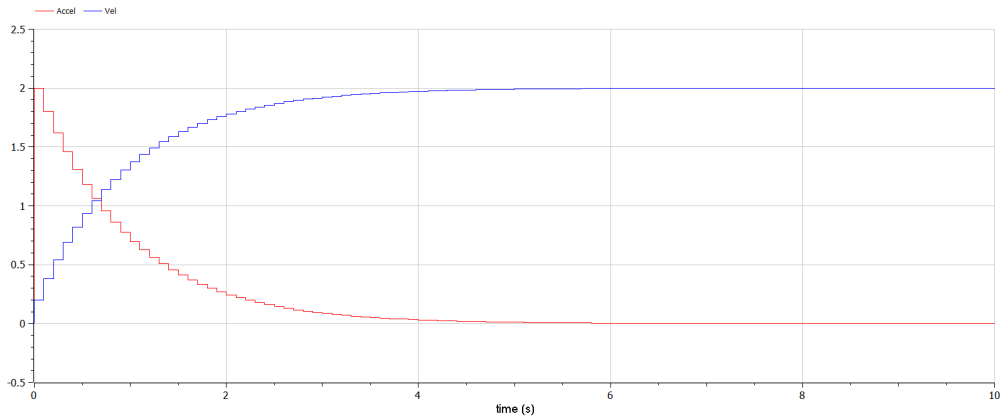

Figure 13: Valores de Velocidad y Aceleración del robot para los parámetros dados.

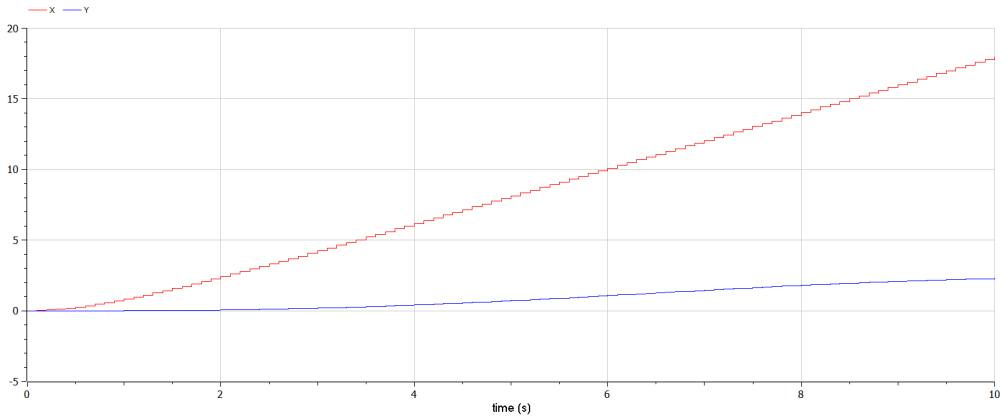

Figure 14: Valores de Posición X e Y del robot para los parámetros dados.

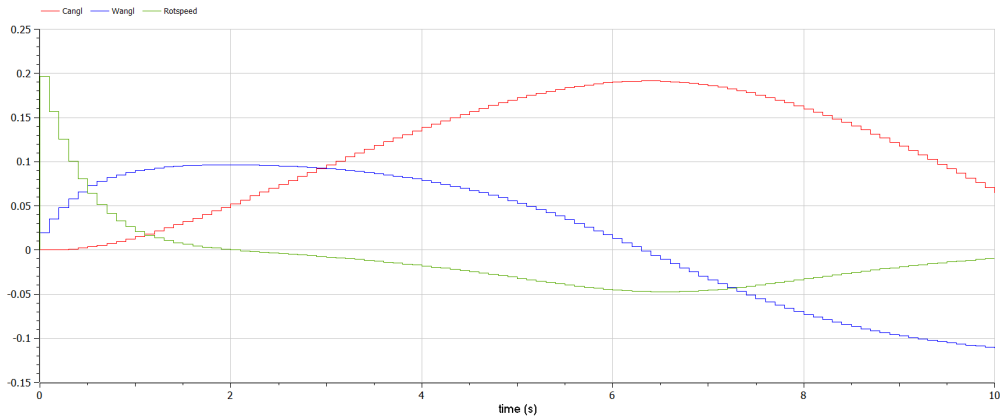

Figure 15: Valores de Angulo del eje del robot y Angulo y Rotación de su dirección para los parámetros dados.

Interpretamos que esta simulación representa la misma situación anterior (un leve desvío y corrección hacia la izquierda) pero esta vez con perfiles mucho más realistas para las curvas generadas por los distintos parámetros del problema (principalmente aquellos referido al ángulo de giro y velocidad del robot).