UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍAS INFORMÁTICA y DE TELECOMUNICACIÓN

Departamento de Ciencias de la Computación e Inteligencia Artificial

SIMULACIÓN DE SISTEMAS

Guión de Prácticas

Práctica 4: Modelos de Simulación Dinámicos Continuos

Curso 2022-2023

Grado en Informática

0.1. Planteamiento

Vamos a estudiar un modelo de propagación de una enfermedad infecciosa (como por ejemplo la covid-19) denominado modelo SIR.

Supongamos una población de N individuos, que podemos dividir en tres grupos:

- Infectados o enfermos, $I(t)$, que tienen la enfermedad y pueden contagiarla.
- Susceptibles o propensos, $S(t)$, que no tienen la enfermedad pero pueden contraerla.
- Retirados, $R(t)$, individuos que se han recuperado de la enfermedad (y ya no la contagian) y se han vuelto inmunes, o que han muerto.

Hacemos las siguientes suposiciones:

1. La población se mantiene constante, es decir, no se tienen en cuenta los nacimientos y muertes (por otras causas) que se producen a lo largo del desarrollo de la enfermedad. Si N es el número de individuos de la población, tenemos pues que $N = S(t) + I(t) + R(t)$.
2. La enfermedad se transmite por contacto directo entre las personas. En cuanto un individuo es infectado pasa a estar en el grupo de los infectados (y contagiosos), es decir no hay periodo de incubación, se pasa inmediatamente del grupo S al grupo I.
3. Los individuos del grupo I se acaban recuperando de la enfermedad y adquieren la inmunidad o mueren, pasando en ambos casos al grupo R. La inmunidad es permanente (ya no vuelven a enfermar).
4. La tasa de infección, es decir el número de individuos por unidad de tiempo que pasan del grupo de susceptibles al de infectados, depende del número de contactos entre individuos susceptibles e infectados, y por tanto es proporcional al producto $S(t) * I(t)$.
5. Los individuos infectados padecerán la enfermedad durante un periodo de tiempo determinado hasta recuperarse y adquirir la inmunidad o morir. La tasa de "recuperación" o retiro es proporcional a $I(t)$.

Tenemos entonces un sistema de tres ecuaciones diferenciales (no lineales):

$$\frac{dI}{dt} = aSI - bI \tag{1}$$
$$\frac{dS}{dt} = -aSI \tag{2}$$
$$\frac{dR}{dt} = bI \tag{3}$$

La velocidad de crecimiento de la población enferma es la diferencia entre la tasa de infección (aSI) y la tasa de recuperación (bI). La constante positiva b está relacionada con el tiempo que dura la infección, es decir, si la enfermedad dura en promedio d días, uno de cada d enfermos en promedio se recupera diariamente, y por tanto $b = 1/d$. La constante positiva a indica la capacidad de infección de la enfermedad, por ejemplo si se estima que es de un 0.1 % del

producto de las poblaciones enferma y propensa (por contactos frecuentes entre miembros de una y otra población), entonces sería $a = 0{,}001$ (expresado en tanto por uno).

Partiendo de unas condiciones iniciales I_0, S_0 y R_0 que representen el número inicial de individuos enfermos, propensos y retirados, se pretende estudiar la evolución en el tiempo de esas poblaciones. Por ejemplo podríamos tener $I_0 = 1$ (enferma un único individuo al principio), $R_0 = 0$ (no hay individuos inmunes) y $S_0 = 999$ (todos los individuos de la población de 1000 excepto uno son susceptibles de enfermar).

Vamos a considerar también (para compararlo con el anterior) otro modelo en que la inmunidad no es permanente, y pasado un cierto tiempo, los individuos inmunes dejan de serlo y se vuelven de nuevo susceptibles. En este caso, el sistema de tres ecuaciones diferenciales es:

$$\frac{dI}{dt} = aSI - bI \tag{4}$$

$$\frac{dS}{dt} = -aSI + cR \tag{5}$$

$$\frac{dR}{dt} = bI - cR \tag{6}$$

donde la constante positiva c está relacionada con la duración de la inmunidad, si ésta dura $dinm$ días, entonces $c = 1/dinm$.

0.2. Tareas

Hacer lo siguiente:

1. Programar un modelo de simulación para estos dos sistemas, de acuerdo a las especificaciones que se indican después.

2. Para unos valores dados de los parámetros, por ejemplo $a = 0{,}001$, $b = 1/8$ y $c = 1/180$ (probad también con otros valores), investigad qué ocurre si el número inicial de individuos susceptibles, S_0, es mayor o menor que b/a. En este punto y en los siguientes excepto el último utilizad el método de integración de Runge-kutta con intervalo de cálculo $h = 0{,}1$.

3. Para hacerse una idea de la evolución del sistema, se pueden hacer representaciones gráficas (superpuestas se observa más claramente) de los valores de I, S y R en función del tiempo. Otra forma interesante de visualizar la evolución es hacer representaciones gráficas de los valores de una población frente a los de otra (es decir, en lugar de emplear los planos t–I y t–S y t–R, usar por ejemplo el plano S–I). ¿Se pueden sacar algunas conclusiones sobre cómo es la evolución en estos sistemas?

4. Probad con distintos valores de los parámetros para observar cómo los sistemas se comportan en diferentes circunstancias. Por ejemplo si se disminuye el valor de a porque disminuyen los contactos (restricciones de movilidad, confinamiento) o se aumenta el valor de b (tratamientos más eficaces disminuyen la duración de la enfermedad), o se modifica el valor de c (haciendo que la inmunidad dure más o menos tiempo).

5. Investigad también los efectos de aumentar el número inicial de individuos infectados, o el número inicial de individuos inmunes (para ver cómo se puede alcanzar la inmunidad de grupo o de rebaño).

6. Comparar los resultados obtenidos por la simulación, cuando se emplea el método de Euler y cuando se usa el de Runge-Kutta. Probad con distintos valores para el intervalo de cálculo (por ejemplo, $h = 0{,}1, 0{,}05, 0{,}01$).

0.3. El programa

El programa de simulación se puede estructurar de la siguiente forma:

- El programa principal únicamente tendrá un procedimiento de captura de parámetros (desde teclado, un fichero,...) que determine los valores de: parámetros a y b, intervalo de cálculo `dt`, intervalo de comunicación, el tiempo inicial `tinic`, tiempo final de simulación `tfin`, y valores iniciales de estado (en nuestro caso los valores para I, S y R, que se suelen almacenar en un array `estado`). Se inicializa el tiempo `t` al valor `tinic` y a continuación se realiza a una llamada al procedimiento de integración `integracion`.

- El procedimiento `integracion` produce en primer lugar la salida (a fichero, monitor,...) en caso de que así sea necesario (en función del intervalo de comunicación, llamando al procedimiento `salida`), después almacena el estado actual `estado` en un estado previo `oldestado`, y llama a un procedimiento `one-step` que realiza un paso del proceso de integración. Finalmente se incrementa el tiempo añadiéndole el valor del intervalo de cálculo. Todo esto se repite hasta que el valor del tiempo supere al valor `tfin`.

- El procedimiento `derivacion` es el encargado de evaluar las ecuaciones de definición del modelo, calculando y devolviendo como salida los valores de las derivadas, `f`. Toma como entradas el estado actual del sistema, `est`, y el valor del tiempo actual, `tt`.

- El procedimiento `one-step` es el que implementa realmente el método de integración numérica. Realiza las llamadas al procedimiento `derivacion`, una única en el caso del método de Euler y cuatro para el método de Runge-Kuta. Toma como entrada el estado actual (almacenado en `oldestado`), el tiempo actual `t` y el valor del intervalo de cálculo `dt`[1], y devuelve el nuevo estado en `estado`.

```
main {
   fijar_parametros();
   t=tinic;
   integracion();
}

procedimiento integracion()
{
   do {
      salida();
      oldestado=estado;
      one-step(oldestado,estado,t,dt);
       t+=dt;
   } while (t<tfin);
}
```

[1]Esto último sólo es necesario si dicho intervalo se puede modificar, es decir, en aquellos casos en que se utilice un método de control de errores basado en disminuir o aumentar el intervalo de forma dinámica.

procedimiento `one-step-runge-kutta(inp,out,tt,hh)`

`inp, out` son vectores de estado: Un array de reales (float o double), con dimensión igual al número de ecuaciones del modelo, `numeq` (3 en nuestro caso particular).
`inp` recibe el estado actual y `out` devuelve el nuevo estado.
`tt` es el tiempo actual y `hh` es el valor del intervalo de cálculo.
`k` es una matriz de dimensión `numeq`$\times 4$ (3×4 en nuestro caso) (o dicho de otra forma, un array de vectores de estado de dimensión 4).
`f` es también un vector de estado (que almacena los valores de las derivadas).
`incr` y `time` son variables reales.
`i` y `j` son variables enteras.

```
{
   for (i=0; i<numeq; i++) out[i]=inp[i];
   time=tt;
   for (j=0; j<4; j++) {
      derivacion(out,f,time);
      for (i=0; i<numeq; i++) k[i,j]=f[i];
      if j<2 incr=hh/2
      else incr=hh;
      time=tt+incr;
      for (i=0; i<numeq; i++) out[i]=inp[i]+k[i,j]*incr;
   }
   for (i=0; i<numeq; i++)
      out[i]=inp[i]+hh/6*(k[i,0]+2*k[i,1]+2*k[i,2]+k[i,3]);
}
```

procedimiento `one-step-euler(inp,out,tt,hh)`

```
{
   derivacion(inp,f,tt);
   for (i=0; i<numeq; i++) out[i]=inp[i]+hh*f[i];
}
```

procedimiento `derivacion(est,f,tt)`
(específico para el modelo considerado)

```
   //Modelo con inmunidad permanente:
{
   f[0]=a*est[0]*est[1] – b*est[0];
   f[1]=–a*est[0]*est[1]
   f[2]=b*est[0];
}
   //Modelo sin inmunidad permanente:
{
   f[0]=a*est[0]*est[1] – b*est[0];
   f[1]=–a*est[0]*est[1] + c*est[2];
   f[2]=b*est[0] – c*est[2];
}
```