SIMULACIÓN DE SISTEMAS

Práctica 4: Modelos de Simulación Dinámicos Continuos
Curso 2019/2020

Considérese el modelo de ecosistema de Lotka-Volterra comentado en clase de teoría, definido por el siguiente sistema de ecuaciones diferenciales (x representa la población de presas e y la de depredadores):

$$\frac{dx}{dt} = a_{11}x - a_{12}xy$$

$$\frac{dy}{dt} = a_{21}xy - a_{22}y$$

Hacer lo siguiente:

1. Programar un modelo de simulación para este sistema, de acuerdo a las especificaciones que se indican después.

2. Para unos valores dados de los parámetros a_{ij} (por ejemplo $a_{11} = 5, a_{12} = 0{,}05, a_{21} = 0{,}0004, a_{22} = 0{,}2$), comprobad qué ocurre cuando las condiciones iniciales son $x = \frac{a_{22}}{a_{21}}$, $y = \frac{a_{11}}{a_{12}}$. Investigad qué ocurre cuando los valores iniciales de x e y se van progresivamente distanciando de los valores anteriores. En este punto y en los dos siguientes 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 x y de y en función del tiempo. Otra forma muy interesante de visualizar la evolución es hacer representaciones gráficas de los valores de y frente a los de x (es decir, en lugar de emplear los planos t–x y t–y, usar el plano x–y). ¿Se pueden sacar algunas conclusiones sobre cómo es la evolución en este sistema?

4. Probad con distintos valores de los parámetros a_{ij}, intentando relacionar los valores de estos parámetros con su significado real, y observando la evolución del sistema. Así, por ejemplo, un incremento en el valor de a_{12} representa que los depredadores son cazadores más eficientes (capturan más presas). Un incremento en a_{11} indica que las presas tienen una tasa de crecimiento más rápida. La disminución de a_{22} representa que los depredadores son más longevos. Finalmente, el valor a_{21} está relacionado con el valor nutricional de las presas, de forma cuanto mayor es este valor, menos presas necesita cada depredador para sobrevivir.

5. 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, $h = 0{,}1, 0{,}05, 0{,}01$).

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_{ij}, 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 x e y, 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` (2 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`×4 (2×4 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)

```
{
   f[0]=a11*est[0]–a12*est[0]*est[1];
   f[1]=a21*est[0]*est[1]–a22*est[1];
}
```