Un caso de modelado y simulación

F. Javier Gil Chica

abril, 2009

Resumen

Este artículo se ofrece como introducción a la asignatura de *Modelado y simulación de sistemas dinámicos*. Un sistema dinámico es aquél que se describe mediante un cierto número de variables cuyos valores son función del tiempo. Las teorías que se ocupan de diversas clases de sistemas dinámicos (así, la Mecánica para sistemas mecánicos, la Electricidad para sistemas eléctricos, etc.) no nos proporcionan directamente estas funciones, sino ecuaciones que expresan la velocidad a la que cambian las variables que describen el sistema. Formalmente, nos proporcionan sistemas de ecuaciones diferenciales que después han de ser integradas para obtener propiamente las funciones que expresan los valores de las variables en función del tiempo. A veces, esta integración puede hacerse de forma analítica. A veces, ha de hacerse de forma numérica.

1. Nuestro sistema

Elegimos como sistema ilustrativo un proyectil de masa m. Deseamos conocer su posición y velocidad en todo instante, y quizás algún dato derivado, como su máximo alcance o máxima altura alcanzada. El proyectil puede modelarse de diversas formas (como una masa inerte lanzada con una velocidad inicial o como proyectil autopropulsado) y ponerse en diversos escenarios: movimiento ideal sin rozamiento y sometido a la aceleración constante de la gravedad, con rozamiento dependiente de la velocidad, con rozamiento que depende de la velocidad y de la densidad del aire, con una densidad del aire que varía de una forma u otra según la altura, lanzado desde una plataforma fija o desde una plataforma en movimiento, sometido o no a los efectos de la rotación del sistema de referencia, etc.

Dependiendo del modelo de proyectil elegido y del escenario las ecuaciones que describen su movimiento tendrán una u otra forma. Si las ecuaciones son sencillas, será fácil integrarlas. En caso contrario, será preciso integrarlas numéricamente. Como resultado de la integración numérica, obtenemos listas de números que hemos de combinar para obtener gráficas. La integración numérica se efectúa mediante un programa. Aquí usaremos el lenguaje C y representaremos las gráficas mediante `gnuplot`. Hay otras opciones: `Matlab`, `Maple`, `Mathematica`...

El procedimiento por tanto será el siguiente:

1. Identificar exactamente qué sistema queremos modelar y simular.
2. Con ayuda de la ciencia que se ocupe de los sistemas del tipo al que pertenezca aquél en que estamos interesados, obtendremos un modelo para su comportamiento. Si esto no es suficiente, tendremos que hacer hipótesis razonables.
3. En general, cuando decimos *modelo* nos estamos refiriendo a un sistema de ecuaciones diferenciales.
4. Si el modelo puede resolverse analíticamente, lo resolveremos así. Si no, lo integraremos numéricamente y representaremos gráficamente el resultado.

Hay otros muchos aspectos que pueden tratarse. Por ejemplo, podemos preguntarnos por la estabilidad del sistema, es decir, por su comportamiento cuando $t \to \infty$. Es evidente que no podemos integrar numéricamente hasta $t \to \infty$ y lo normal es que tampoco sepamos integrar analíticamente el modelo, por lo que se requerirán técnicas especiales que se tratan durante el curso. También es posible que no nos baste conocer cómo se va a comportar el sistema, sino que deseemos manipularlo de alguna manera para que se comporte de alguna forma útil para nosotros. Existen igualmente técnicas potentes para conseguir esto que se tratarán también durante el curso.

2. Un método de integración

El sistema dinámico más sencillo que podamos concebir es aquél que se describe mediante una única variable x. Con ayuda de hipótesis o mediante alguna ciencia particular (Mecánica, Electricidad, Química, etc.), obtendremos un modelo dinámico que se expresará como una ecuación diferencial de la forma

$$\dot{x} = f(x,t) \tag{1}$$

$\dot{x}$ indica derivada de x respecto al tiempo. La función $f(x,t)$ puede ser más o menos sencilla, lineal o no lineal, dependiente de x y t o de sólo una de ellas, etc. El problema que se plantea es obtener la función $x(t)$. Existen multitud de algoritmos para la resolución de este tipo de ecuaciones. Nosotros presentaremos uno de ellos, el método de Runge-Kutta de cuarto orden. Es un método popular, sencillo de programar, relativamente rápido y en la mayor parte de las ocasiones de suficiente precisión. No vamos a discutir la forma en que se obtiene dicho método, sino simplemente describir el algoritmo. Antes de eso, conviene aclarar la naturaleza de todos estos métodos. Una integración numérica no nos proporciona la función $x(t)$, sino una serie de valores de x para instantes $t_1, t_2, t_3 ... t_n$. Aunque existen métodos de *paso variable* donde los intervalos $t_{i+1} - t_i$ dependen de i, en la mayoría de las ocasiones, para todos los i, la diferencia $t_{i+1} - t_i$ es una constante h, de manera que si partimos desde un instante inicial t_0, $t_i = t_0 + ih$. Todos estos métodos requieren el conocimiento de un estado inicial a partir del cual calcular estados sucesivos. Particularizando en nuestro sistema, es obvio que la posición y velocidad del proyectil en un instante dado depende de la posición y velocidad con que fue lanzado.

El método de Runge-Kutta parte de un valor x_k (indicamos así $x(t_k)$) y calcula el valor x_{k+1}. Si fijamos h (por ejemplo, $h = 0{,}1$ cuando deseamos conocer x de décima en décima de segundo) el algoritmo sigue los siguientes pasos:

1. Calcular $k_1 = hf(x_k, t_k)$
2. Calcular $k_2 = hf(x_k + k_1/2, t_k + h/2)$
3. Calcular $k_3 = hf(x_k + k_2/2, t_k + h/2)$
4. Calcular $k_4 = hf(x_k + k_3, t_k + h)$
5. Calcular $x_{k+1} = x_k + \frac{1}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right]$

En general, los sistemas dinámicos vienen descritos no por una sola variable, sino por varias. Por consiguiente, necesitamos generalizar el algoritmo para el caso en que queremos integrar no una ecuación sino un conjunto de ellas. Daremos el algoritmo explícito para un sistema de dos ecuaciones. La generalización para un número cualquiera es inmediata y se deja como ejercicio al lector. Sea el sistema:

$$\begin{aligned} \dot{x} &= f(x,y,t) \\ \dot{y} &= g(x,y,t) \end{aligned} \tag{2}$$

Dados unos valores x_k e y_k, el algoritmo sigue los siguientes pasos:

1. Calcular $k_1 = hf(x_k, y_k, t_k)$
2. Calcular $n_1 = hg(x_k, y_k, t_k)$
3. Calcular $k_2 = hf(x_k + k_1/2, y_k + n_1/2, t_k + h/2)$
4. Calcular $n_2 = hg(x_k + k_1/2, y_k + n_1/2, t_k + h/2)$
5. Calcular $k_3 = hf(x_k + k_2/2, y_k + n_2/2, t_k + h/2)$
6. Calcular $n_3 = hg(x_k + k_2/2, y_k + n_2/2, t_k + h/2)$
7. Calcular $k_4 = hf(x_k + k_3, y_k + n_3, t_k + h)$
8. Calcular $n_4 = hg(x_k + k_3, y_k + n_3, t_k + h)$
9. Calcular $x_{k+1} = x_k + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$
10. Calcular $y_{k+1} = y_k + \frac{1}{6}[n_1 + 2n_2 + 2n_3 + n_4]$

3. Variables de estado

Sucede con frecuencia que nos disponemos de modelos que se expresen como sistemas de ecuaciones diferenciales de primer orden, sino de orden dos o superiores. Es el caso de los problemas de la mecánica. Por ejemplo, de la ecuación de Newton:

$$\bar{F} = m\bar{a} \tag{3}$$

Escribiéndola en sus componentes:

$$\begin{aligned} \ddot{x} &= \frac{1}{m}F_x \\ \ddot{y} &= \frac{1}{m}F_y \\ \ddot{z} &= \frac{1}{m}F_z \end{aligned} \tag{4}$$

Este es un sistema de ecuaciones diferenciales de segundo orden, ya que aparecen las segundas derivadas de las variables. Un sistema de este tipo puede siempre reducirse a otro sistema de ecuaciones de primer orden introduciendo *variables de estado*. Sea por ejemplo un modelo de la forma $\ddot{x} = f(x,t)$. Si introducimos variables x_1 y x_2 de forma que

$$\begin{aligned} x_1 &= x \\ x_2 &= \dot{x} \end{aligned} \tag{5}$$

es claro entonces que

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x_1,t) \end{aligned} \tag{6}$$

Otro ejemplo: para un modelo de la forma $x^{(3)} = f(x,t)$, donde $x^{(3)}$ representa la tercera derivada respecto al tiempo de x, introducimos tres variables de estado: $x_1 = x$, $x_2 = \dot{x}$ y $x_3 = \ddot{x}$, con lo cual:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= f(x_1,t) \end{aligned} \tag{7}$$

Hay un caso particular interesante, y es cuando el modelo consiste en una única ecuación diferencial ordinaria de grado n de la forma

$$\frac{d^n x}{dt^n} + a_{n-1}\frac{d^{n-1}x}{dt^{n-1}} + a_{n-2}\frac{d^{n-2}x}{dt^{n-2}} + \cdots + a_1\frac{dx}{dt} + a_0 x = 0 \tag{8}$$

Si introducimos variables de estado

$$\begin{aligned} x_1 &= x \\ x_2 &= \dot{x} \\ x_3 &= \ddot{x} \\ ... &= ... \\ x_n &= \frac{d^{n-1}x}{dt^{n-1}} \end{aligned} \tag{9}$$

es claro que

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= x_4 \\ ... &= ... \\ \dot{x}_n &= -a_0 x_1 - a_1 x_2 - a_2 x_3 - \cdots - a_{n-1} x_n \end{aligned} \tag{10}$$

4. Primer modelo

Nuestro primer modelo para el proyectil es muy simple: una masa m que se lanza con una velocidad inicial desde el origen de coordenadas, sin rozamiento con el aire, en un campo de gravedad constante. Puesto que la gravedad actúa sólo en la dirección del eje z, el movimiento resultante está contenido en el plano (x, z). El proyectil es lanzado en la dirección positiva del eje x y el eje z está dirigido verticalmente hacia arriba. La ecuación de Newton, resuelta en sus componentes, nos dice que la aceleración según el eje x es nula, y que la aceleración según el eje z es $-g$, siendo g el módulo de la aceleración de la gravedad:

$$\begin{aligned} \ddot{x} &= 0 \\ \ddot{z} &= -g \end{aligned} \tag{11}$$

Necesitamos introducir las varibles de estado: $x_1 = x$, $x_2 = z$, $x_3 = \dot{x}$ y $x_4 = \dot{z}$, con lo cual

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= 0 \\ \dot{x}_4 &= -g \end{aligned} \tag{12}$$

Si llamamos $\bar{x}$ al vector de estado de componentes

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{13}$$

el sistema anterior puede escribir en la forma matricial compacta siguiente:

$$\dot{\bar{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \bar{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix} \tag{14}$$

Este es un caso particular de un tipo de sistemas de la forma general

$$\dot{\bar{x}} = \mathbf{A}\bar{x} + \mathbf{B}\bar{u} \tag{15}$$

donde $\mathbf{A}$ y $\mathbf{B}$ son ciertas matrices. Existe una solución analítica para los sistemas de este tipo y de ella se tratará durante el curso. Este sistema se integra de forma trivial, de manera que podemos obtener expresiones analíticas para $x(t)$ e $y(t)$, y de ahí obtener algunos resultados de interés, como el máximo alcance, altura máxima, ecuación de la envolvente, tiempo de vuelo, etc. Estos resultados se encuentran en los textos elementales, así que no los discutiremos aquí. En lugar de eso, implementaremos un programa que integre numéricamente el modelo. Seguiremos los pasos desde la resolución numérica hasta la obtención de una gráfica. Después, usaremos esa gráfica para comparar este primer modelo con modelos más elaborados.

Puesto que con frecuencia la visualización de los datos es relevante, nos detendremos en los detalles del proceso que lleva desde la resolución numérica hasta la generación del documento final que incluye una gráfica. Por supuesto, existen muchas formas de hacer esto, y seguramente más cómodas que la que adoptamos aquí.

El siguiente es el listado del programa que implementa el algoritmo de Runge-Kutta para resolver el sistema de ecuaciones (12) [1].

[1]Por conveniencia, hemos suprimido la indentación original, para que el código fuente pueda adaptarse al ancho de este documento. No será un grave inconveniente, puesto que el programa es corto

```
#include <stdio.h>
#include <math.h>

#define pi 3.14159265

#define G 9.81   /* aceleracion gravedad */
#define V0 100   /* velocidad inicial */

double k[4],l[4],m[4],n[4];
double h=0.01;
double t=0;

double x0[4], x1[4];

FILE *listado;


double f0(double x0, double x1, double x2, double x3, double t)
{
    return(x2);
}

double f1(double x0, double x1, double x2, double x3, double t)
{
    return(x3);
}

double f2(double x0, double x1, double x2, double x3, double t)
{
    return(0);
}

double f3(double x0, double x1, double x2, double x3, double t)
{
    return(-G);
}

main()
{

if ((listado=fopen("datos.dat","w"))==NULL) return;

/* establecer condiciones iniciales */
x0[0]=0;
x0[1]=0;
x0[2]=V0*cos(pi/4);
x0[3]=V0*sin(pi/4);
```

```
/* mientras la coordenada z sea >=0 */
while (x0[1]>=0){

/* salvar a archivo */
fprintf(listado,"%f %f %f %f %f\n",t,x0[0],x0[1],x0[2],x0[3]);

/* paso 1 */
k[0]=h*f0(x0[0],x0[1],x0[2],x0[3],t);
l[0]=h*f1(x0[0],x0[1],x0[2],x0[3],t);
m[0]=h*f2(x0[0],x0[1],x0[2],x0[3],t);
n[0]=h*f3(x0[0],x0[1],x0[2],x0[3],t);

/* paso 2 */
k[1]=h*f0(x0[0]+k[0]/2,x0[1]+l[0]/2,x0[2]+m[0]/2,x0[3]+n[0]/2,t+h/2);
l[1]=h*f1(x0[0]+k[0]/2,x0[1]+l[0]/2,x0[2]+m[0]/2,x0[3]+n[0]/2,t+h/2);
m[1]=h*f2(x0[0]+k[0]/2,x0[1]+l[0]/2,x0[2]+m[0]/2,x0[3]+n[0]/2,t+h/2);
n[1]=h*f3(x0[0]+k[0]/2,x0[1]+l[0]/2,x0[2]+m[0]/2,x0[3]+n[0]/2,t+h/2);

/* paso 3 */
k[2]=h*f0(x0[0]+k[1]/2,x0[1]+l[1]/2,x0[2]+m[1]/2,x0[3]+n[1]/2,t+h/2);
l[2]=h*f1(x0[0]+k[1]/2,x0[1]+l[1]/2,x0[2]+m[1]/2,x0[3]+n[1]/2,t+h/2);
m[2]=h*f2(x0[0]+k[1]/2,x0[1]+l[1]/2,x0[2]+m[1]/2,x0[3]+n[1]/2,t+h/2);
n[2]=h*f3(x0[0]+k[1]/2,x0[1]+l[1]/2,x0[2]+m[1]/2,x0[3]+n[1]/2,t+h/2);

/* paso 4 */
k[3]=h*f0(x0[0]+k[2],x0[1]+l[2],x0[2]+m[2],x0[3]+n[2],t+h);
l[3]=h*f1(x0[0]+k[2],x0[1]+l[2],x0[2]+m[2],x0[3]+n[2],t+h);
m[3]=h*f2(x0[0]+k[2],x0[1]+l[2],x0[2]+m[2],x0[3]+n[2],t+h);
n[3]=h*f3(x0[0]+k[2],x0[1]+l[2],x0[2]+m[2],x0[3]+n[2],t+h);

/* nuevos valores */
x1[0]=x0[0]+(1.0/6.0)*(k[0]+2*(k[1]+k[2])+k[3]);
x1[1]=x0[1]+(1.0/6.0)*(l[0]+2*(l[1]+l[2])+l[3]);
x1[2]=x0[2]+(1.0/6.0)*(m[0]+2*(m[1]+m[2])+m[3]);
x1[3]=x0[3]+(1.0/6.0)*(n[0]+2*(n[1]+n[2])+n[3]);

/* actualizar */
x0[0]=x1[0];
x0[1]=x1[1];
x0[2]=x1[2];
x0[3]=x1[3];
t=t+h;

} /* end while */

fclose(listado);
return;
}
```

Hemos tomado un incremento $h = 0{,}01$. El proyectil es lanzado desde el origen de coordenadas con una velocidad de 100 m/s y formando un ángulo de $\pi/4$ con el eje horizontal. El programa calcula mientras el proyectil se encuentre sobre el suelo, es decir, mientras sea $z >= 0$. Los datos se guardan en el archivo `datos.dat`, que es un archivo de texto que contiene cinco columnas, a saber: el tiempo, la coordenada x, la coordenada z, la velocidad según el eje x y la velocidad según el eje z.

Hemos usado el programa `gnuplot` para representar gráficamente la trayectoria del proyectil, es decir, para representar la gráfica $z(x)$. La gráfica puede representarse en pantalla (la opción por defecto) pero puede también guardarse en otros formatos. Hemos elegido PostScript encapsulado porque podemos incorporar esos gráficos a nuestro documento `.tex` mediante el paquete `dvips`. Una vez dentro de `gnuplot`, escribimos

```
> set terminal fig
> set output "trayectoria-0.fig"
> set xlabel "x"
> set ylabel "z(x)"
> set grid
> plot "datos.dat" using 2:3 with dots notitle
```

La primera línea indica al programa que la salida no será a pantalla, sino en el formato propio del programa `xfig`. La segunda línea establece el nombre del archivo gráfico. Las líneas tercera y cuarta establecen las etiquetas para los ejes horizontal y vertical. La quinta línea fuerza a que se dibuje la cuadrícula, que por defecto se omite. Una vez hecho esto, la última línea es la que se encarga efectivamente de trazar la gráfica, mediante la orden `plot`, que toma los parámetros siguientes: `datos.dat` es el nombre del archivo donde se encuentran los datos; `using 2:3` indica a `plot` que, de todas las columnas que componen el archivo, use la segunda y la tercera; la segunda contiene los valores de x y la tercera los valores de z; `with dots` indica a `plot` que no trace una línea continua, uniendo puntos, sino que cada pareja (x, z) sea representada por un punto; `notitle` indica a `plot` que omita en la gráfica una etiqueta que aparece por defecto con el nombre del archivo de donde provienen los datos.

Una vez generado el archivo gráfico, podemos editarlo y modificarlo si fuese preciso desde `xfig` y a continuación lo exportamos como archivo `.eps` y procedemos a incluirlo en nuestro documento fuente `.tex`. Para ello, introducimos las siguientes líneas:

```
\begin{figure}{h}
\begin{center}
\includegraphics[scale=1.0]{trayectoria-0.eps}\\
{\bf Figura 1}
\end{center}
\end{figure}
```

El resultado es el que aparece:

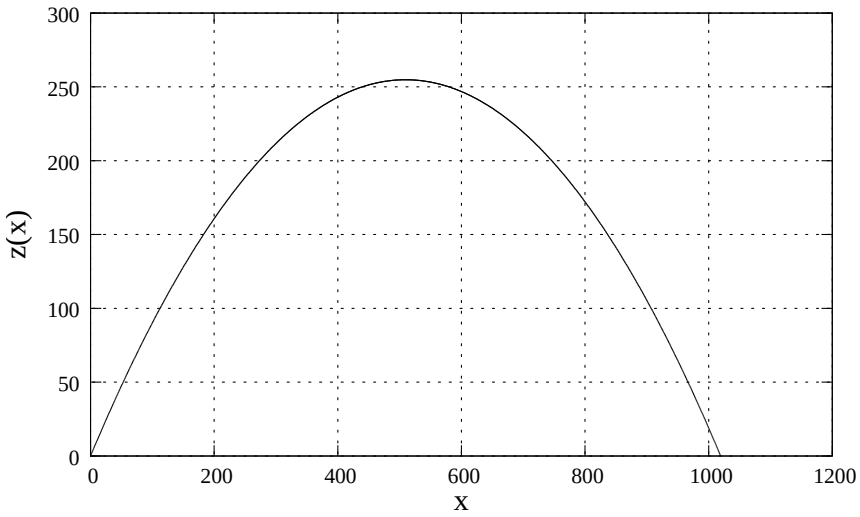

Figura 1

5. Segundo modelo

Modelaremos el movimiento del proyectil suponiendo que existe un rozamiento proporcional al cuadrado del módulo de la velocidad y de sentido contrario a la misma. Por consiguiente, la ecuación de Newton incluye tanto la fuerza de la gravedad como la resistencia del medio:

$$m\bar{a} = m\bar{g} - kv\bar{v} \tag{16}$$

Simplificando la masa, introduciendo la constante $k' = k/m$ y escribiendo expresiones explícitas para las componentes x y z tenemos que

$$\begin{aligned} \ddot{x} &= -k'v\dot{x} \\ \ddot{z} &= -k'v\dot{z} \end{aligned} \tag{17}$$

donde

$$|v| = \sqrt{\dot{x}^2 + \dot{z}^2} \tag{18}$$

Introduciendo variables de estado $x_1 = x$, $x_2 = z$, $x_3 = \dot{x}$ y $x_4 = \dot{z}$:

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= -k'vx_3 \\ \dot{x}_4 &= -g - k'vx_4 \end{aligned} \tag{19}$$

Obsérvese que, respecto al programa presentado anteriormente, sólo hemos de modificar el cuerpo de las funciones `f0()`, `f1()`, `f2()` y `f3()`. El resto queda igual. En la Figura 2 puede apreciarse el efecto del rozamiento sobre el alcance y la máxima altura, para los valores $k' = 0$ (caso sin rozamiento), $k' = 0{,}0001$ y $k' = 0{,}001$:

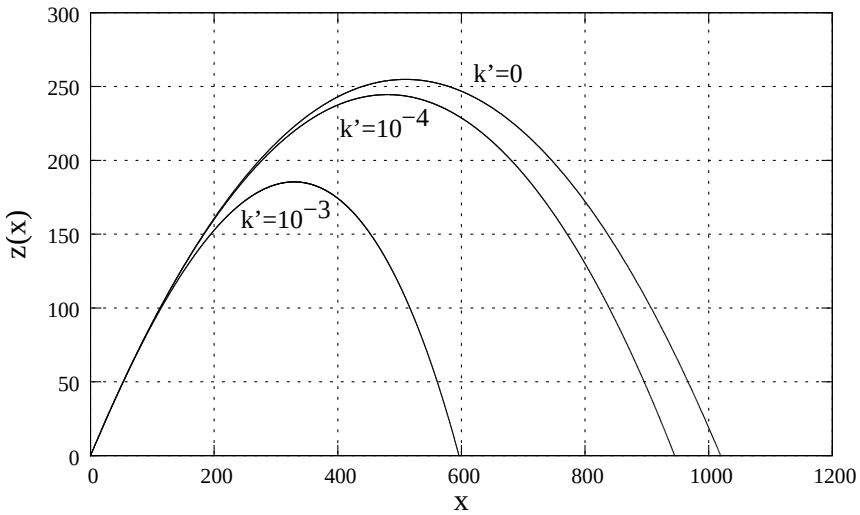

Figura 2

6. Tercer modelo

Hemos considerado en el modelo anterior que el movimiento del proyectil sufre una fuerza de rozamiento que depende del cuadrado de la velocidad a la que se mueve. También depende de la densidad del medio en el que se mueve. Si esta densidad es constante, podemos

considerarla integrada en la constante k', de forma que $k' = c\rho$, siendo c otra constante. Pero sabemos que la densidad del aire varía con la altura, de forma que $k' = c\rho(z)$. Hemos de buscar una forma para la función $\rho(z)$. Podríamos acudir a la hidrostática pero, como no deseamos entrar en ese punto, haremos alguna hipótesis razonable.

Sabemos que la densidad es máxima en la superficie y que decrece progresivamente (no hay "borde" en la atmósfera) hasta que desaparece prácticamente a unos 100 km (el límite es algo arbitrario). Una función que se adapta a estas características es una exponencial:

$$\rho(z) = \rho(0)\exp(-\alpha z) \tag{20}$$

Queda pendiente el ajuste de las constantes. Para ajustar α, usaremos el hecho de que la densidad del aire se reduce a la mitad aproximadamente a unos 5000 m. de altura:

$$\rho(5000) = \frac{1}{2}\rho(0) = \rho(0)\exp(-\alpha 5000) \tag{21}$$

de donde

$$\alpha = -\frac{1}{5000}\ln(\frac{1}{2}) \sim 1{,}386 \times 10^{-4} \tag{22}$$

Para ajustar c, podemos hacerlo conjuntamente con $\rho(0)$, de forma que $k' = c\rho(0)$ y coincida numéricamente con el valor de k' usado en el modelo anterior. De esta forma, la simulación de este modelo requiere unicamente la sustitución de k' por $k'\exp(-\alpha z)$. Por otro lado, el efecto será tanto más evidente cuanta mayor sea la altura alcanzada por el proyectil. Para conseguir mayor altura, elevamos el ángulo de tiro desde $\pi/4$ a $3\pi/8$ e incrementamos la velocidad desde los 100 m/s hasta los 1000 m/s. Repetimos la simulación del segundo modelo con los nuevos valores de ángulo y velocidad y comparamos con el modelo actual. El resultado de la simulación se presenta en la Figura 3

7. Cuarto modelo

Nos planteamos ahora el movimiento de un proyectil autopropulsado. Un proyectil de tal clase ha de llevar consigo el combustible. Ese combustible se consume a un ritmo determinado y el empuje resultante es proporcional a ese ritmo. A medida que se consume el combustible, disminuye la masa del proyectil, de forma que ya no puede considerarse ésta constante. El modelo por tanto ha de partir de

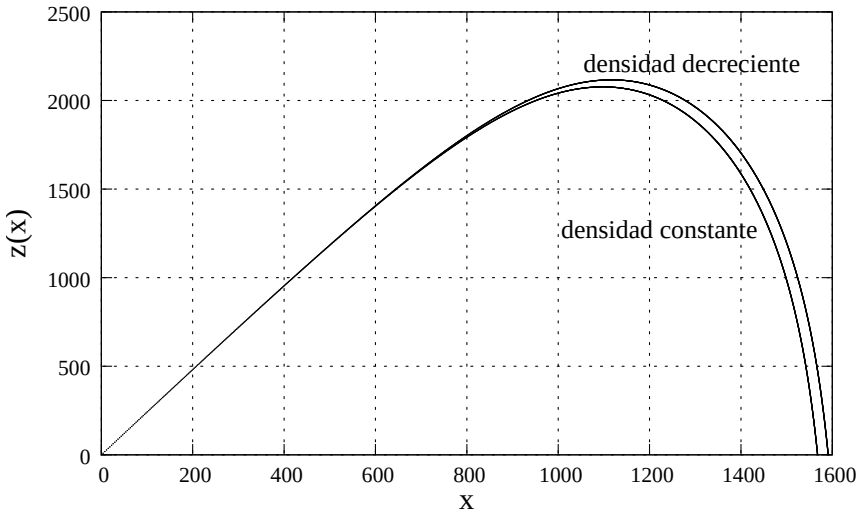

Figura 3

alguna formulación más general que la forma básica de la ecuación de Newton $\bar{F} = m\bar{a}$. No es preciso ir muy lejos:

$$\bar{F} = \frac{d}{dt}\bar{p} \tag{23}$$

con $\bar{p} = m(t)\bar{v}$. Si el ritmo al que se quema combustible es constante:

$$m(t) = m(0) - \beta t \tag{24}$$

tenemos

$$\dot{\bar{p}} = -\beta\bar{v} + m(t)\dot{\bar{v}} \tag{25}$$

Como fuerzas, consideraremos la fuerza de la gravedad, la fuerza de rozamiento proporcional al cuadrado de la velocidad y la fuerza de autopropulsión, tangente a la trayectoria en todo punto:

$$\bar{F} = \bar{g} - k'v\bar{v} + \gamma\beta\frac{\bar{v}}{v} \tag{26}$$

Y éste es el modelo:

$$\bar{g} - k'v\bar{v} + \gamma\beta\frac{\bar{v}}{v} = -\beta\bar{v} + m(t)\dot{\bar{v}} \tag{27}$$

Introduciremos algunos parámetros descriptivos. En primer lugar, si la masa inicial es $m(0)$, llamaremos f a la fracción de esa masa que es combustible, y τ al tiempo durante el cual ese combustible va a proporcionar empuje. En cuanto a ese empuje, lo consideraremos constante. De aquí:

$$m(\tau) = (1-f)m(0) = m(0) - \beta\tau \tag{28}$$

de donde

$$\beta = \frac{f}{\tau}m(0) \tag{29}$$

Por otra parte, el módulo del empuje ha de ser superior al peso del proyectil [2], luego

$$\gamma\beta > m(0)g \tag{30}$$

de donde

$$\gamma > \frac{\tau g}{f} \tag{31}$$

Introduciendo el factor $p > 1$ escribimos:

$$\gamma = p\frac{\tau g}{f} \tag{32}$$

Con estas definiciones, escribimos el modelo en la forma:

$$\bar{g} - kv\bar{v} + pm(0)g\frac{\bar{v}}{v} = -\frac{f}{\tau}m(0)\bar{v} + m(0)(1-\frac{f}{\tau}t)\dot{\bar{v}} \tag{33}$$

Que se desdobla en las componentes:

$$\begin{aligned} -kv\dot{x} + pm(0)g\frac{\dot{x}}{v} &= -\frac{f}{\tau}m(0)\dot{x} + m(0)t'\ddot{x} \\ -m(0)gt' - kv\dot{z} + pm(0)g\frac{\dot{z}}{v} &= -\frac{f}{\tau}m(0)\dot{z} + m(0)t'\ddot{z} \end{aligned} \tag{34}$$

donde por aligerar la escritura hemos hecho

$$t' = 1 - \frac{f}{\tau}t \tag{35}$$

[2]En rigor, no es así, pero prescindimos de los detalles

Introduciendo variables de estado tenemos:

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= \left[-\frac{kv}{m(0)t'} + \frac{pg}{vt'} + \frac{f}{\tau t'}\right] x_3 \\
\dot{x}_4 &= -g + \left[-\frac{kv}{m(0)t'} + \frac{pg}{vt'} + \frac{f}{\tau t'}\right] x_4
\end{aligned} \tag{36}$$

Hemos realizado algunas simulaciones de este sistema, con valores $m = 100kg.$, $p = 3$, $f = 0{,}6$, $\tau = 200s$ y $v(0) = 10m/s$. De nuevo, el programa que presentamos en el primer modelo sólo precisa modificaciones en las definiciones de las funciones `f0()`, `f1()`, `f2()` y `f3()`. Hay algunos aspectos cualitativos nuevos. En primer lugar, es preciso comprobar en cada paso de integración si queda aún combustible o no. El combustible se agota cuando $t = \tau$. En segundo lugar, el proyectil no es "disparado", sino que abandona una rampa de lanzamiento por sus propios medios. La velocidad a la salida de la rampa es comparativamente baja, y por tanto comparativamente a un proyectil no autopropulsado del mismo alcance el ángulo de salida ha de ser más elevado. La Figura 4 presenta la trayectoria para dos ángulos distintos: 45^o y 50^o.

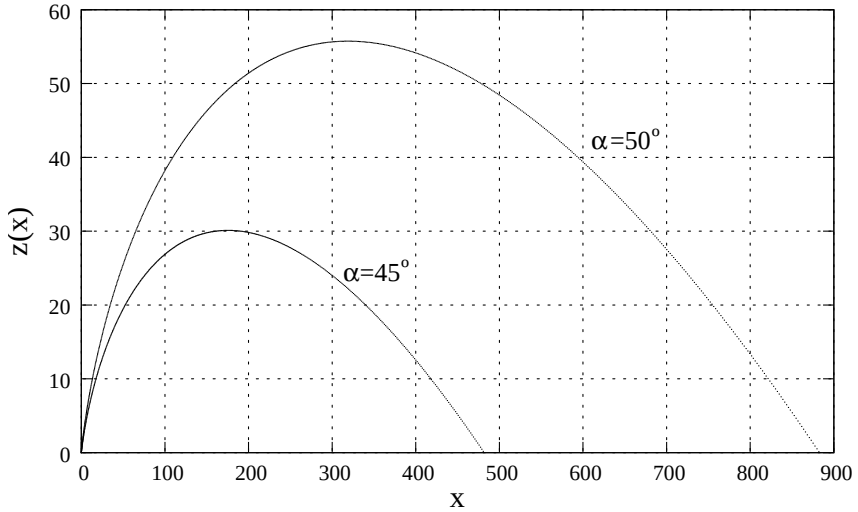

Figura 4

Vemos que el alcance es sensible respecto al ángulo de salida. De hecho, esa diferencia de 5^o prácticamente duplica el alcance. Más to-

davía, con un ángulo de 80^o el alcance sube hasta los 210 km. y la altura máxima que se alcanza es de aproximadamente 21000 m.

En tercer lugar, para la altura alcanzada es ya muy apreciable la disminución de la densidad del aire, y por tanto de la resistencia. Este factor no lo hemos tenido en cuenta en esta simulación.

En cuarto lugar, para un alcance de más de 200 km. ya es apreciable el efecto de la curvatura de la Tierra, y de su rotación.

En quinto lugar, es preciso examinar el tiempo de vuelo. Para los ángulos de salida de 45^o y 50^o estos tiempos son aproximadamente de 6 y 8 s., y eso significa que sólo una pequeña fracción del combustible se ha gastado, lo cual no tiene sentido. Para el ángulo más elevado de 80^o el tiempo de vuelo es de 160 s., quedando aún una fracción no despreciable de combustible por consumir. Efectuamos entonces otra simulación con un ángulo de salida de 85^o y comprobamos cómo el alcance y la altura máxima se incrementan, tal y como se muestra en la Figura 5:

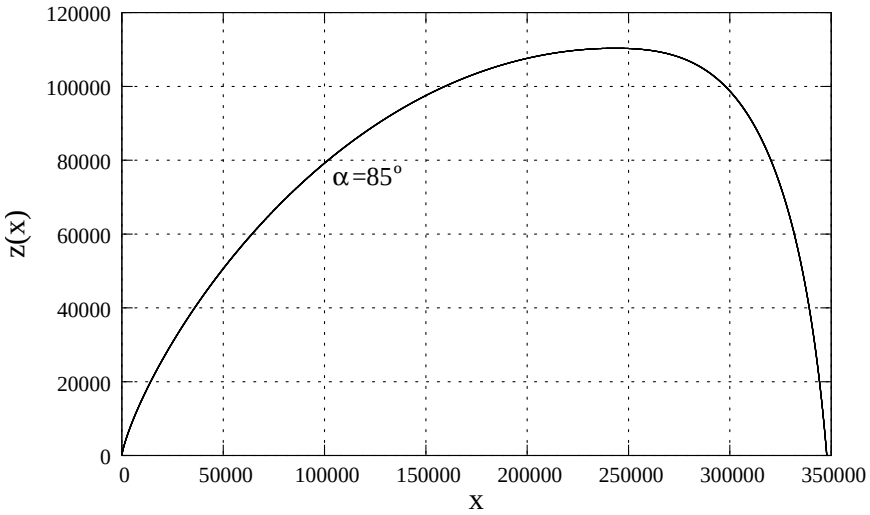

Figura 5

Es también interesante examinar la curva $x(t)$, que se muestra en la Figura 6:

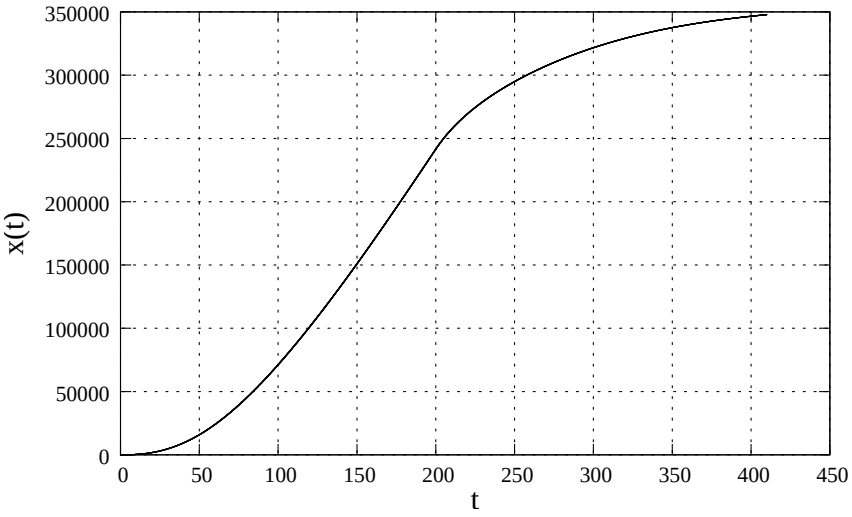

Figura 6

Como se ve, el tiempo de vuelo supera los 400 s. y la curva es cóncava hasta $t = 200s.$, indicando la aceleración del motor, pasando a convexa a partir de ese punto, indicando la deceleración por rozamiento. Este efecto se aprecia con mucha mayor claridad si se examina la gráfica para la velocidad en función del tiempo, representada en la Figura 7:

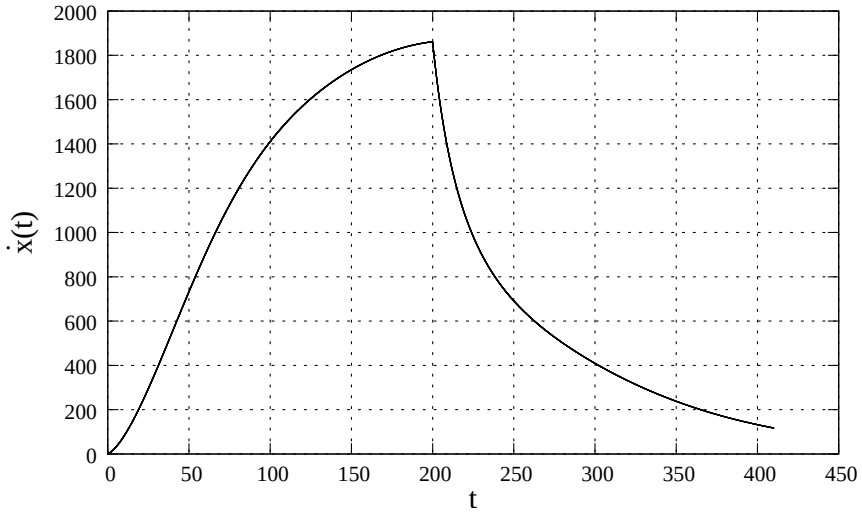

Figura 7

También es interesante examinar la gráfica para la velocidad vertical, Figura 8:

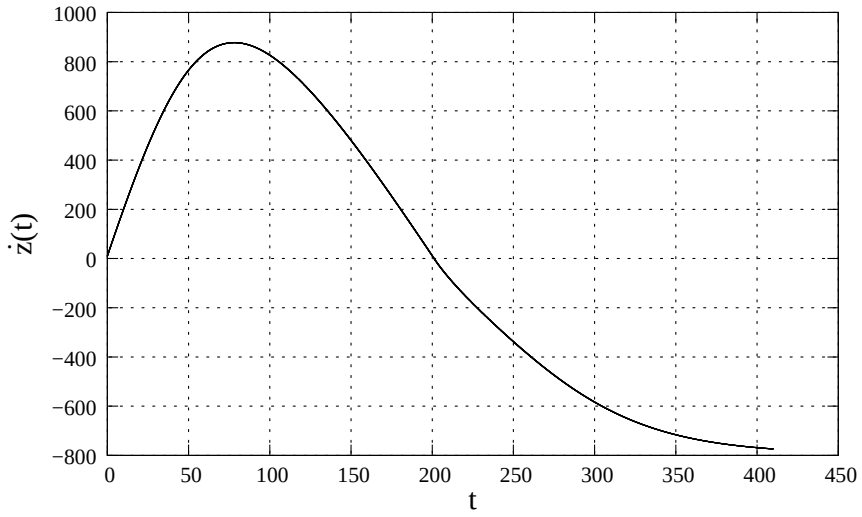

Figura 8

8. Quinto modelo

Las ecuaciones de Newton son válidas para sistemas de referencia inerciales. Pero cuando un proyectil es lanzado desde tierra, lo hace desde una plataforma giratoria, sometida por tanto a aceleración, es decir, desde un sistema no inercial en el que no son válidas las ecuaciones de Newton. Sea (x, y, z) un sistema inercial y (u, v, w) un sistema móvil que comparte origen con el primero. Los ejes z y w coinciden y el sistema (u, v, w) gira con velocidad angular constante θ alrededor del eje z.

Entre ambos sistemas existe la relación geométrica:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{37}$$

Si consideramos el movimiento de un proyectil sometido sólo a la fuerza de la gravedad, en el sistema inercial sus ecuaciones del movimiento son:

$$\begin{aligned} \ddot{x} &= 0 \\ \ddot{y} &= 0 \\ \ddot{z} &= -g \end{aligned} \tag{38}$$

Es una cuestión simplemente algebraica escribir estas ecuaciones en función de las coordenadas del sistema móvil:

$$\begin{aligned} \ddot{u} - 2\dot{\theta}\dot{v} - \dot{\theta}^2 u &= 0 \\ \ddot{v} + 2\dot{\theta}\dot{u} - \dot{\theta}^2 v &= 0 \\ \ddot{w} &= -g \end{aligned} \tag{39}$$

Introduciendo variables de estado $x_1 = u$, $x_2 = v$, $x_3 = w$, $x_4 = \dot{u}$, $x_5 = \dot{v}$ y $x_6 = \dot{w}$:

$$\begin{aligned} \dot{x}_1 &= x_4 \\ \dot{x}_2 &= x_5 \\ \dot{x}_3 &= x_6 \\ \dot{x}_4 &= \dot{\theta}^2 x_1 + 2\dot{\theta} x_5 \\ \dot{x}_5 &= \dot{\theta}^2 x_2 - 2\dot{\theta} x_4 \\ \dot{x}_6 &= -g \end{aligned} \tag{40}$$

Aparentemente, tenemos un sistema de seis ecuaciones, pero la tercera y la sexta no dependen de ninguna de las otras: se encuentran *desacopladas* y por tanto pueden resolverse de forma independiente la primera, segunda, cuarta y quinta por un lado, y la tercera y sexta por otro. Tercera y sexta tienen solución trivial:

$$\begin{aligned} x_6(t) &= \dot{z} = x_6(0) - gt = \dot{z}(0) - gt \\ x_3(t) &= x_3(0) + x_6(0)t - \frac{1}{2}gt^2 = z(0) + \dot{z}(0)t - \frac{1}{2}gt^2 \end{aligned} \tag{41}$$

de donde obtenemos el tiempo de vuelo:

$$t = \frac{2\dot{z}(0)}{g} \tag{42}$$

Integraremos durante este intervalo el sistema formado por las ecuaciones restantes. Como dijimos antes, si el sistema es lineal existe una solución analítica y no es preciso hacer simulación alguna. Pero aquí no estamos en ningún momento intentando resolver los sistemas que vamos encontrando: nuestro interés está en el modelado y simulación.

Hemos simulado dos disparos, a lo largo del eje u del sistema de referencia móvil (fijo respecto a la plataforma, que gira), uno en sentido positivo y otro en sentido negativo. La Figura 9 representa la proyección de la trayectoria sobre el plano (u, v). El resultado es que, con una velocidad inicial de 100 m/s y un ángulo de lanzamiento de $\pi/4$, para un alcance de unos 1000 m. se produce una desviación de aproximadamente 2.5 m. a la derecha, tomando como velocidad angular la mitad de la velocidad angular de rotación de la Tierra:

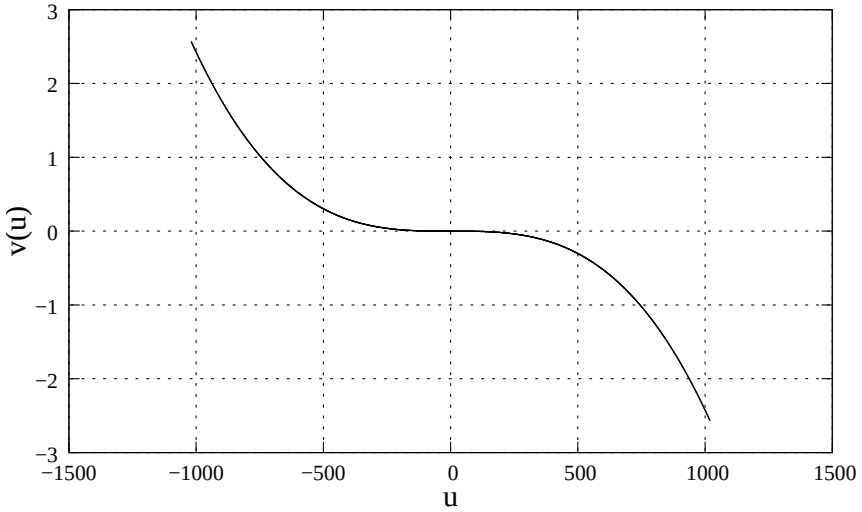

Figura 9

9. Conclusiones

Hemos usado la Física elemental para plantear cinco versiones de un mismo sistema: un proyectil que se mueve bajo la acción de la gravedad. No hemos resuelto analíticamente ninguno de los problemas planteados, sino que hemos escrito algunos programas, implementando el algoritmo de Runge-Kutta de cuarto orden, para simularlos. Hemos prestado algo de atención al proceso de representación de los resultados, desde el archivo de datos en bruto generado por el programa de simulación hasta la inserción de una gráfica en un documento `.pdf` (este documento).

Se observará que algunas veces la Física es suficiente para generar un modelo. Otras, será preciso hacer hipótesis razonables. En cualquier caso, hemos mostrado un método susceptible de ser aplicado a

multitud de problemas dinámicos, entendidos éstos en el sentido amplio de sistemas descritos por variables que varían en el tiempo, no necesariamente sistemas mecánicos.

Al mismo tiempo, esperamos haber sugerido otras preguntas, de las que dejamos constancia de unas cuantas: ¿cómo modelar efectos aerodinámicos, como por ejemplo la sustentación del proyectil? ¿qué ocurre cuando se ha de tener en cuenta la curvatura de la Tierra? ¿Qué ocurre si un misil se aleja lo bastante de la superficie de la Tierra como para tener que considerar que la aceleración de la gravedad varía tanto en módulo como en dirección?

Hay un amplísimo campo que ni siquiera hemos sugerido, aunque forma parte del curso: el control de sistemas. Relativas al control del sistema propuesto, algunas de las preguntas que podríamos hacer son: ¿cómo conseguir que el proyectil impacte en una posición predeterminada? ¿cómo conseguir que siga una trayectoria prefijada? ¿cómo conseguir que el misil auto-corrija el efecto de perturbaciones atmosféricas o errores en alguno de los parámetros de lanzamiento? En general ¿cómo conseguir que la trayectoria del misil satisfaga algún criterio prefijado?

10. Apéndice

En la página 9 se puede ver el código del algoritmo de Runge-Kutta de cuarto orden aplicado al caso de un sistema de cuatro ecuaciones diferenciales ordinarias. A la vista de dicho código, es fácil ver que la ampliación del algoritmo a un problema con un número arbitrariamente mayor de ecuaciones se reduce (desde el punto de vista de la codificación) a *cortar y pegar*.

Esto motiva la pregunta de si sería posible escribir una versión de dicho algoritmo que tomase como parámetro el número de ecuaciones. También, si sería posible aislar el algoritmo de las funciones concretas de los segundos miembros del sistema de ecuaciones diferenciales.

La respuesta a ambas preguntas es sí, y en este apéndice presentamos una implementación del algoritmo que toma como parámetros esencialmente el número de ecuaciones y un vector de punteros a función. Además, un puntero a un vector con las componentes del estado del sistema, los instantes inicial y final, el intervalo de integración y un archivo donde se guardarán el tiempo y el estado del sistema en cada

instante. El algoritmo se presenta en la función `rk4()` y como puede verse en el programa de ejemplo que se adjunta el funcionamiento típico consiste en: a) escribir las funciones de los segundos miembros (`f0,1,2,3()` en nuestro caso); b) Establecer condiciones iniciales; c) asignar los punteros a función y d) ejecutar `rk4()`

```
/*
Este programa ilustra el uso de la funcion rk4,
que es un algoritmo generico de Runge-Kutta de
cuarto orden que recibe como parametros el orden
del sistema, un array de punteros a funcion,
un vector de estado, instantes inicial y final y
paso de integracion. Tambien un archivo, donde
se escribiran en cada paso el valor del tiempo
y las componentes del vector de estado.

Los punteros a funcion apuntan a funciones que
toman como parametros un puntero al vector de
estado y quizas el tiempo.

Seria mas modular escribir una funcion que calculase
solo un paso de integracion, y usarla como base para
una segunda que en un bucle efectuara la integracion
completa. Pero como esta primera funcion tendria que
reservar y liberar memoria en cada llamada, seria
poco eficiente. Asi que escribimos en una unica
funcion el bucle completo y luego, en cada caso, si
es preciso anyadimos dentro de ese bucle lo que
necesitemos.
*/


#include <stdio.h>
#include <math.h>
#include <malloc.h>

#define pi 3.14159265

/*
devuelve 1 si todo bien; 0 si no se pudo abrir el
archivo donde se guardarn los datos
*/
```

```
int rk4(int n, double (*g[])(double*, double), double *x,
         double t0, double t1, double h, char *archivo)
{

    int j;

    /* las k's: n filas por 4 columnas */
    double *k=(double *)malloc(n*4*sizeof(double));

    /* el estado siguiente */
    double *y=(double *)malloc(n*sizeof(double));

    /* la variable tiempo */
    double t=t0;

    /* el archivo para los datos */
    FILE *f;

    if ((f=fopen(archivo,"w"))==NULL) return(0);

    /* el bucle: aqui puede ponerse la condicion que se quiera */
    while (t<t1-h){

        /* paso cero: guardar tiempo y estado */
        fprintf(f,"%10.4f ",t);
        for(j=0;j<n;++j)
            fprintf(f,"%10.4f ",*(x+j));
        fprintf(f,"\n");

        /* primer paso */
        for(j=0;j<n;++j)
            *(k+4*j)=h*(*g[j])(x,t);

        /* segundo paso: preparar argumento y llamar */
        for(j=0;j<n;++j)
            *(y+j)=(*(x+j)) + 0.5* (*(k+4*j));
        for(j=0;j<n;++j)
            *(k+4*j+1)=h*(*g[j])(y,t+0.5*h);

        /* tercer paso: preparar argumento y llamar */
        for(j=0;j<n;++j)
            *(y+j)=(*(x+j)) + 0.5* (*(k+4*j+1));
        for(j=0;j<n;++j)
```

```
            *(k+4*j+2)=h*(*g[j])(y,t+0.5*h);

        /* cuarto paso: preparar argumento y llamar */
        for(j=0;j<n;++j)
            *(y+j)=(*(x+j)) + (*(k+4*j+2));
        for(j=0;j<n;++j)
            *(k+4*j+3)=h*(*g[j])(y,t+h);

        /* quinto paso: calcular nuevo estado */
        for(j=0;j<n;++j)
            *(x+j)=(*(x+j))+(1.0/6.0)*( (*(k+4*j)) + 2* (*(k+4*j+1))
                     + 2* (*(k+4*j+2)) + (*(k+4*j+3)) );

        /* sexto paso: actualizar el tiempo */
        t+=h;

    }

    /* cerrar el archivo */
    fclose(f);

    /* liberar memoria */
    free(k);
    free(y);

    return(1);
}

/*
ejemplo de calculo de trayectorias de proyectiles
pr.pdf, pag. 6
*/
double f0(double *x, double t){
    return(*(x+2));
}
double f1(double *x, double t){
    return(*(x+3));
}
double f2(double *x, double t){
    return(0);
}
double f3(double *x, double t){
    return(-9.81);
```

```
}

main()
{
    int s;

    double (*g[4])(double *, double);

    /* valores iniciales */
    double *x=(double *)malloc(4*sizeof(double));

    *(x+0)=0;
    *(x+1)=0;
    *(x+2)=100*cos(pi/4);
    *(x+3)=100*sin(pi/4);

    g[0]=f0;
    g[1]=f1;
    g[2]=f2;
    g[3]=f3;

    s=rk4(4,g,x,0,10,0.1,"prueba.dat");
    if (s) printf("\nOK\n"); else printf("\nNOK\n");

    return;
}
```